TIETO- JA SÄHKÖTEKNIIKAN TIEDEKUNTA

ELEKTRONIIKAN JA TIETOLIIKENNETEKNIIKAN TUTKINTO-OHJELMA

KANDIDAATINTYÖ

XILINXIN 7-SARJAN FPGA-RESURSSIT ZYNQ-JÄRJESTELMÄPIIRILLE

Tekijä Sami Tervonen

Ohjaaja Jukka Lahti

Lokakuu 2021

Tervonen S. (2021) Xilinxin 7-sarjan FPGA-resurssit Zynq-järjestelmäpiirille. Oulun yliopisto, Tieto- ja sähkötekniikan tiedekunta, Elektroniikan ja tietoliikennetekniikan tutkinto-ohjelma. Kandidaatintyö, 46 s.

TIIVISTELMÄ

Tässä kandidaatintyössä esitellään Zynq-järjestelmäpiiri ja sen yleisrakenne. Pääpaino on kuitenkin sen ohjelmoitavan logiikan resursseissa, jotka ovat Xilinxin 7-sarjaa. Näiden resurssien osalta käydään läpi tarkemmin muistiin, digitaaliseen signaalinprosessointiin sekä suorittavaan logiikkaan liittyviä lohkoja.

Avainsanat: ASIC, FPGA, Järjestelmäpiiri, SoC, APSoC, FPGA-resurssi, Spartan-7, Artix-7, Kintex-7, Virtex-7, CLB, DSP48E1, RAMB36E1, SelectIO, GTX/GTH, XADC, PCIe.

Tervonen S. (2021) Xilinx 7-series FPGA resources for Zynq-SoC. University of Oulu, Faculty of Information and Electrical Engineering, Degree Programme of Electrical and Information Technology, Bachelor's Thesis, 46 s.

ABSTRACT

This Bachelor's Thesis is about a SoC (System-on-a-Chip) called Zynq. The basic structure of this chip is presented but the main focus is in its Xilinx 7-series resources located on the Programmable Logic. Resources like memory block, digital signalling processing block and configurable logic block are presented in detail.

Key words: ASIC, FPGA, System-On-Chip, SoC, APSoC, FPGA-resource, Spartan-7, Artix-7, Kintex-7, Virtex-7, CLB, DSP48E1, RAMB36E1, SelectIO, GTX/GTH, XADC, PCIe.

SISÄLLYSLUETTELO

ALKULAUSE

Kauas on tultu siitä, kun Digitaalitekniikka 2 -kurssin tentin suoritin. Harjoitustyö muhi takaraivossa lähes 14 vuotta kunnes sain sen suoritettua v. 2016. Suurin kiitos tästä kuuluu Jukka Lahdelle, joka jaksoi vastailla lukuisiin kysymyksiin kurssin harjoitustyön aikana. En olisi koskaan uskonut, että digitaalielektroniikka kiinnostaisi binäärisiä lukuja enempää. Tässä kuitenkin ollaan!

Oulussa 16.10.2021

Sami Tervonen

LYHENTEIDEN SELITYKSET

SoC	System-on-Chip. Järjestelmäpiiri, missä kaikki järjestelmän tarvitsemat komponentit on sijoitettuna samalle piikiekolle.
ASIC	Application-Specific Integrated Cirquit. Asiakkaan tarpeisiin kustomoitu IC-piiri. Piirille ei voida sen kasaamisen jälkeen tehdä rautapohjaisia muutoksia.
FPGA	Field Programmable Gate Array. Piikiekolle sijoitettu, yleensä MOS-transistoreista koottu verkko, jonka voi ohjelmoida sovelluksen vaatimalla tavalla.
IP	Intellectual Property. Yleensä käytetään nimitystä valmiiksi suunnitellusta lohkosta, joka toteuttaa halutun toiminnon, esimerkiksi FIR-suodatin.
CPU	Central Processing Unit. Keskussuoritinyksikkö, joka hoitaa pääohjemien suorituksen ja tehtävien jakamisen oheislaitteille.
GPU	Graphics Processing Unit. Keskussuoritinyksikkö graafisten toimintojen toteuttamiseen. Hoitaa yleensä median suoritukseen liittyvät tehtävät.
AXI	Advanced eXtensible Interface. Kättelymenetelmään perustuva väyläarkkitehtuuri, jota käytetään yleisesti Zynq-piireissä ohjelmoitavan logiikan ja suoritinpuolen välillä.
DSP	Digital Signal Processing. Digitaalinen signaalinprosessointi
PS	Processing System. Suoritinpuoli, jonka ARM Cortex-A9 suoritinytimet oheismuisteineen ja -laitteineen hoitavat koko Zynq-piirin päätoiminnan.
PL	Programmable Logic. Ohjelmoitava logiikka eli Zynqin tapauksessa FPGA-verkko, jonka rautapohjaiset ohjelmoidut lohkot toimivat apuna ARM Cortex-A9 suoritinytimille.
APU	Application Processing Unit. Ohjelmansuoritusyksikkö, jossa ARM Cortex-A9 fyysisesti sijaitsevat. Hoitaa ohjelmien varsinaisen suorituksen.
NEON	Ydin, joka auttaa ARM Cortex-A9 prosessoriytimiä rinnakkaisessa laskennassa (media).
CLB	Configurable Logic Block. Ohjelmoitavalle logiikalle sijoitettu logiikkalohko, joka voidaan ohjelmoida halutulla tavalla. Voidaan valjastaa myös muistiksi (RAM/ROM), siirtorekisteriksi tai valintamultiplekseriksi.
FIFO	First-In First-Out. RAM-muisti, jolla ei ole ulospäin näkyviä muistiosoitteita ja josta luetaan ja johon kirjoitetaan sisäisten laskureiden osoittamassa järjestyksessä.
BRAM	Block Random Access Memory. Max 36kB dataa varastoiva hajasaantimuistilohko, joka sijaitsee ohjelmoitavalla logiikalla.
2-komplementti	Kahden komplementti. Binääriluvun esitys, missä binääriluvun bitit käännetään ympäri ja lisätään yksi. Tällöin vähennyslaskun sijaan kahden luvun bitit vain lisätään yhteen ja saadaan haluttu vähennyslaskun tulos.
MUX	MUltipleXer. Valintamultiplekseri, jolla voidaan valita kahdesta tai useammasta tulosta lähtöön (lähtöihin) menevät(t) signaali(t).
LUT	Look-Up Table. Hakutaulukko eli funktiogeneraattori, joka voi suorittaa loogisia toimintoja kuten AND, OR, XOR, NAND jne.
IOB	Input/Output Block. Lohko, joka puskuroi sille menevää/siltä tulevaa dataa ja tekee mm. sovitukset niin, että signaalit eivät heijastu tai huoju.
DCI	Digitally Controlled Impedance. Lähetinvastaanotinten tulo- ja lähdeimpedanssien säätö digitaalisesti.
GT	Gigabit Transceiver. Usean GB/s nopeudella toimiva lähetinvastaanotinrakenne ohjelmoitavalla logiikalla.
XADC	Xilinxin oma analogia-digitaalimuunnin-rakenne ohjelmoitavalla logiikalla.
CMT	Clock Management Tile. Hoitaa esim. kelloon lukittautumisen mm. sisäisen vaihelukitun silmukan avulla.

1. JOHDANTO

Tämä kandidaatintyö käsittelee järjestelmäpiiriä (engl. System-on-Chip, SoC) nimeltä Zynq, joka on Xilinxin valmistama tuote. Siinä yhdistyvät ARM:n kaksiytiminen Cortex A9 -suoritin sekä Xilinxin 7-sarjan FPGA-verkko. Työn tarkoituksena on selvittää, mitä FPGA-resursseja Xilinx tarjoaa Zynq-piirille ja avata hieman näiden resurssien rakenteita ja perusominaisuuksia elektroniikkasuunnittelun näkökulmasta. Se, miksi tämän kyseisen piirin resursseja lähdin yleensäkin tutkimaan, on Zynqistä löytyvä kattava dokumentaatio ja tuki. Lisäksi Zynqin ympärillä on jatkuva yliopistollinen tutkimus, joten sille löytyy kattava määrä valmiiksi suunniteltuja kokonaisuuksia (engl. Intellectual Property, IP).

Ennen kuin mennään Zynq-piirin tarkempaan rakenteeseen ja resursseihin, avataan hieman yleistä käsitystä järjestelmäpiiristä, sen suunnittelusta ja optimointiasteista.

1.1 Järjestelmäpiiri ja Zynq

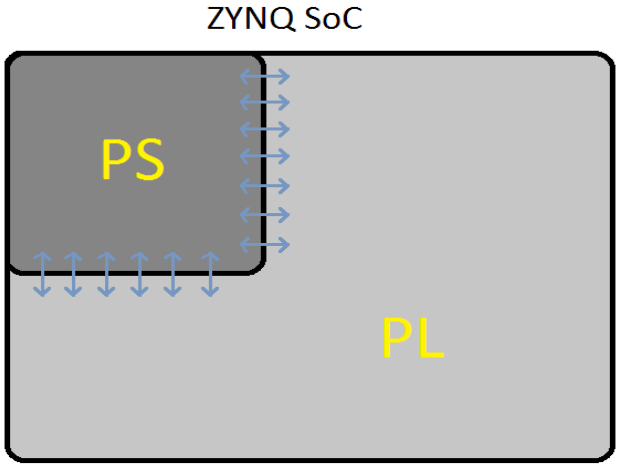
Järjestelmäpiiri on piiri (engl. chip), johon on yhdistettynä kaikki yhdelle samalle piikiekolle: keskussuoritinyksikkö (engl. Central Processing Unit, CPU), muistia (ROM, RAM), graafinen suoritinyksikkö (engl. Graphics Processing Unit, GPU), analogia-digitaalimuuntimet, lähetinvastaanottimet tiedonsiirtoa varten jne [1].

Järjestelmäpiiri on todella suuri kokonaisuus, jolla voidaan ohjata mm. älytalon valaistusta, ilmanvaihtoa, kulunvalvontaa siihen liitettyjen tunnistinten ja anturoiden avulla. Järjestelmäpiiri on usein osana sulautettua järjestelmää, mm. pelikonsoleissa (nykyajan PlayStation, XBOX jne.). Niitä löytyy myös tutuista arkipäiväisistä laitteista, kuten älypuhelimista ja tableteista, joissa järjestelmäpiirin parhaimmat puolet kuten pieni tehonkulutus ja ala, hyvästä suorituskyvystä puhumattakaan, pääsevät ominaisuuksiinsa.

Edellä mainitut kaupalliset SoC:t ovat ASIC-piirejä (engl. Application-Specific Integrated Cirquit) eli asiakkaalle kustomoituja IC-piirejä: ne on tarkasti optimoitu juuri tiettyyn tehtävään sopiviksi ja niille ei voi myöhemmin tehdä muutoksia. Muutoksia halutessaan piirin valmistajan täytyy suunnitella, optimoida, testata ja valmistaa uusi piiri. Tämä on usein tuskallisen hidas prosessi, ja siksi pelkkää ASICia ei kannata lähtökohtaisesti valita, mikäli järjestelmältä halutaan joustavuutta ja piirin rakennetta pitää myöhemmin muuttaa.

ASICia joustavamman ratkaisun tarjoaa FPGA (engl. Field-Programmable Gate Array) eli siinä transistoriverkkoa voidaan ohjelmoida halutun laiseksi ja saada näin ollen sinne juuri sovellukseen sopiva kokoonpano. Se on kuitenkin hieman rajoittunut, sillä FPGA-verkko vie melkoisen alan ja lisäksi se on ASIC-piirejä hitaampi. Joustavamman ja lisäksi tehokkaamman ratkaisun tarjoaa näiden yhdistelmä eli jo aiemmin mainittu FPGA-SoC. Siinä prosessori (tai mikrokontrolleri) välimuisteineen muodostaa nk. *suoritinpuolen* (engl. Processing System, PS) ja FPGA-verkkoa kutsutaan *ohjelmoitavaksi logiikaksi* (engl. Programmable Logic, PL).

Suoritinpuolelle asennetaan mm. käyttöjärjestelmä, jonka päällä ohjelmat, eli appit, toimivat. Ohjelmoitavalle logiikalle voidaan taas ohjelmoida erilaisia rautapohjaisia oheislaitteita, jotka toimivat lisäapuna prosessorille. Tällaisia oheislaitteita ovat apuprosessorit (MicroBlaze), muistit, digitaalisen signaalinprosessoinnin yksiköt (engl. Digital Signal Processing Unit, DSP Unit), GPU sekä muut systeemin tarvitsemat komponentit. Tässä työssä käsiteltävä Zynq on juuri tällainen piiri, jota Xilinx mainostaa lisäksi APSoC:ina (All Programmable SoC) eli siinä kaikki voidaan ohjelmoida käyttäjän haluamalla tavalla. Kuva 1 havainnollistaa Zynqin yksinkertaistettua rakennetta.



Kuva 1. Zynqin yksinkertaistettu rakenne. Vasemmalla suoritinpuoli (Processing System, PS), jonka ympärillä ohjelmoitava logiikka (Programmable Logic, PL)

1.2 Järjestelmäpiirin suunnittelusta ja optimointiasteista

Nykyään digitaalisen elektroniikan suunnittelu on viety yhä enenemässä määrin korkeammalle suunnittelun tasolle. Tämä tarkoittaa, että siirrytään rekisterisiirtotason kovonkuvauksesta ylemmän tason kuvaukseen, joka toteutetaan pääosin C++/SystemC-kielellä. Tällöin ei vältämättä tarvita erillistä SystemVerilog/VHDL-osaajaa vaan suunnittelu voidaan toteuttaa perehtymättä komponentin rakenteellisiin ominaisuuksiin. Nykyajan korkean abstraktiotason synteesiohjelmat (engl. High Level Synthesis, HLS) ovat kehittyneet jo niin pitkälle, että niillä voidaan nopeasti liittää valmiiksi suunniteltuja komponentteja osaksi suunnitelmaa ja näin ollen madaltaa valmiin tuotteen hintaa ja markkinoille pääsyä. HLS-ohjelmilla suunnittelijan ei välttämättä tarvitse suunnitella peruslohkoja itse, vaan mm. Xilinxiltä löytyy kirjastoistaan suuri määrä standardeja malleja testipenkkeineen ja simulointeineen. Nämä IP-mallit on helppo integroida omaan suunnitelmaan ja mm. optimointi ja testaus on suoraviivaista tehdä ohjelmiston tukemien simulointi- ja optimointityökalujen avulla. Zynqin tapauksessa myös kolmansilta osapuolilta löytyy kattava määrä erilaisia IP-malleja.

Kirjallisuudessa ja yleensä digitaalielektroniikkaan liittyvässä suunnittelussa törmää väistämättä sanaan ydin (engl. core). Sanana tämä on varmasti monelle tuttu nykyajan prosessoriytimistä puhuttaessa, mutta digitaalielektroniikassa sana ydin on yleisnimitys suunnitellulle lohkolle. Zynqin suoritinpuolelta löytyvät ARM Cortex 2 -prosessoriytimet ovat erittäin pitkälle optimoituja ja testattuja eli nk. kovia (hard) ytimiä. Optimointiasteita on yhteensä kolme: soft/firm/hard core, missä ”pehmeä” ydin on suunnittelun löyhin aste ja optimointeja ei ole tehty, kun taas kovasta ytimestä on luotu ajoitus/testausmallit ja layout-tiedostot. Sanomattakin on selvää, että kovan ytimen IP-mallin saadessaan suunnittelija ei voi enää muuttaa SoC:n tarvitsemaa alaa, pienentää tehonkulutusta tai kasvattaa suorituskykyä, sillä rankat optimointirajat menevät tällöin kokonaan uusiksi ja piiri todennäköisemmin ei enää toimi lainkaan.

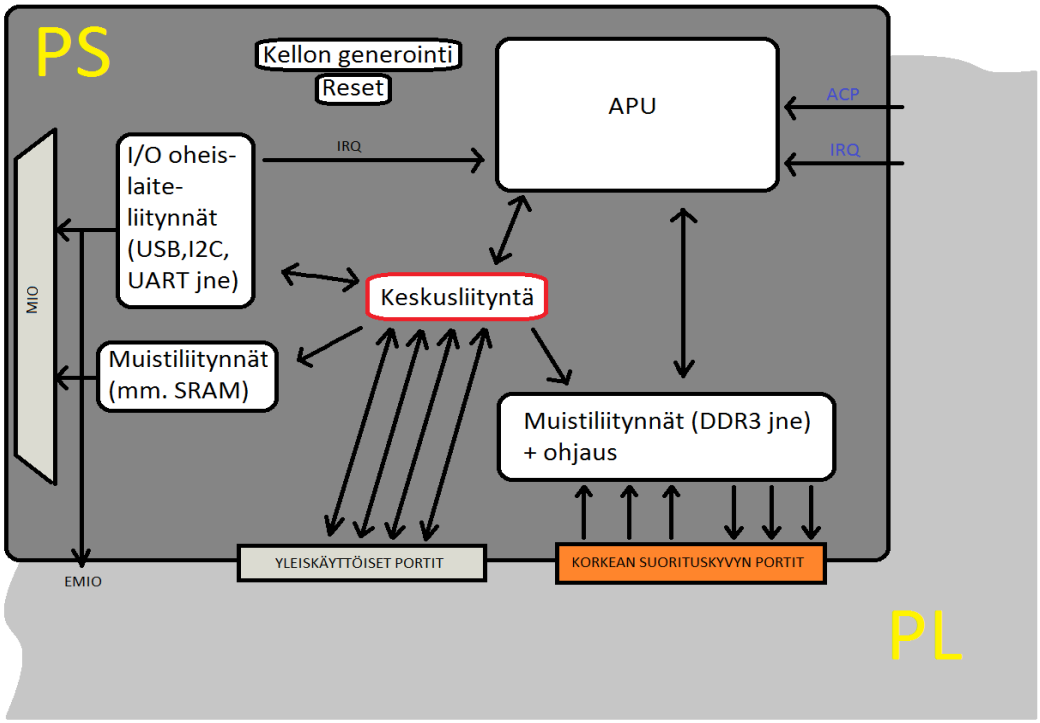
Järjestelmäpiireille olemassa olevat IP-mallit on hyvä jättää optimoimatta, jotta seuraavalla suunnittelijalla olisi vapaammat kädet integroitaessa kyseinen IP osaksi suunnitelmaansa. IP kannattaa myös tehdä mahdollisimman hyvin uudelleenkäytettäväksi eli pikkutarkan koodin kommentoinnin ja kattavan dokumentoinnin avulla käyttäjä pääsee nopeasti jyvälle lohkon toiminnasta. Ennen IP:n integroimista omaan suunnitelmaan, IP on kuitenkin syytä aina simuloida ja tarkistaa ensin itsenäisesti sillä IP:n alkuperäinen suunnittelija ei ole mitenkään voinut tietää kaikkia tulevia toimintaympäristöjä.

Lisää tietoa järjestelmäpiirien suunnittelusta ja testauksesta löytyy lähdeluettelon kohdasta [2].

2. ZYNQ-JÄRJESTELMÄPIIRIN RAKENNE JA OMINAISUUDET

2.1 SUORITINPUOLI

Kuten edellä jo mainittiin, koostuu FPGA-SoC suoritinpuolesta ja ohjelmoitavasta logiikasta. Suoritinpuolelle käyttäjä voi asentaa halutessaan oman Linux-käyttöjärjestelmän ja tämän käyttöjärjestelmän päällä suoritettavat ohjelmat eli appit ajetaan. Zynq on itsessään sulautettu järjestelmä, jonka suunnittelu alkaa jakamalla toiminnalliset lohkot softa- tai kovo-puoleen (raakajaottelu ARM Cortexilla ajettavat ohjelmat vs. FPGA-verkon rauta). Softa- ja kovopuoli suunnitellaan ja optimoidaan ensin erikseen ja sitten kovo-softa-lohkot testataan yhdessä ja integroidaan yhteen toimivaksi järjestelmäksi. Zynqissä suoritinpuolen ja ohjelmoitavan logiikkapuolen (PS-PL) välillä toimii nopea AXI (engl. Advanced eXtensible Interface)-väylä, jota voidaan käyttää korkean suorituskyvyn ja korkean taajuuden järjestelmissä ja se on osana ARM:n AMBA:n avointa standardia. Rakenne on laajasti käytössä sen yksinkertaisen rakenteen vuoksi ja sen kättelymenetelmä master-slave-välillä mahdollistaa melko suoraviivaisen tiedonsiirron. Kuvassa 2 on havainnollistettu suoritinpuolta (PS).

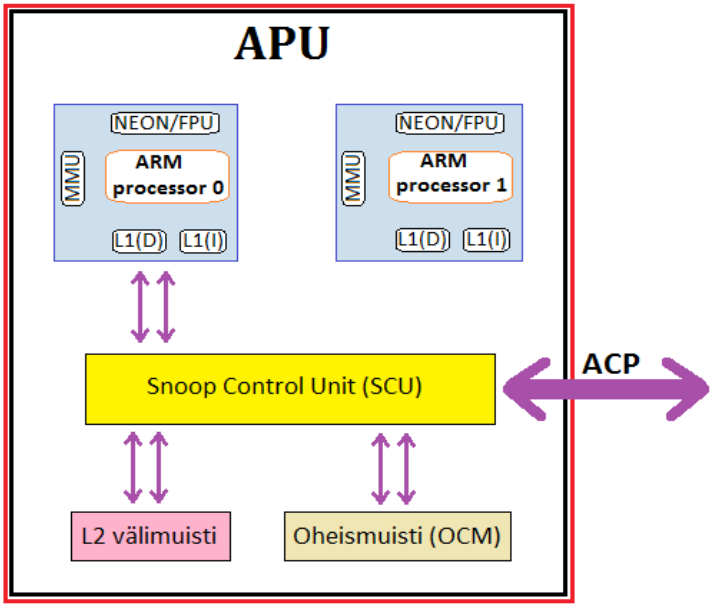


Kuva 2. Zynqin suoritinpuoli (Processing System, PS).

Suoritinpuolen komponentteja ovat: ohjelmansuoritusyksikkö (engl. Application Processor Unit, APU), kellon generointi- ja resetointipiiri sekä erilaiset liitynnät. Lii-tyntöjä taas ovat muisti-, ja oheislaiteliitynnät sekä keskusliityntä, jonka kautta APU voi olla yhteydessä liityntöihin ja tämän kautta ulkomaailmaan. Avataan seuraavissa kappaleissa liityntöjä ja ohjelmansuoritusyksikön rakennetta.

2.1.1 Ohjelmansuoritusyksikkö (APU)

Tärkein komponentti Zynqin suoritinpuolella on ohjelmansuoritusyksikkö (engl. APU, Application Processor Unit), joka sisältää kaksiytimisen ARM:n Cortex A9 -prosessorin välimuisteineen ja oheislaitteineen. Se toimittaa koko Zynqin ohjelmalliset toiminnot ja ottaa vastaan tietoa ulkomaailman tuloporteista ja ohjaa lähtöportteja ulkomaailmaan. APU:n rakenne on esitettynä kuvassa 3.



Kuva 3. Ohjelmansuoritusyksikkö, APU

Oheislaitteista NEON-ydin sekä SCU on syytä käydä hieman tarkemmin läpi, mutta muut lohkot ja oheislaitteet voidaan esittää yksinkertaisemmin. Näitä ovat:

- ARM processor 0 / processor 1
  - ydin, joka hoitaa ohjelmalliset laskennat
- L1(x)
  - välimuisti, jossa sijaitsee eniten käytetyt käskyt ja data
  - L1(D) välimuisti datalle, koko 32kB
  - L1(I) välimuisti käskyille, koko 32kB
- L2
  - prosessoriydinten jakama yhteinen suurempi lisävälimuisti, koko 512kB
- OCM
  - On-Chip-Memory, APU:n läheisyydessä sijaitseva lisämuisti, koko 256kB
- MMU
  - Memory Management Unit, hoitaa virtuaalisten osoitteiden käännöt fyysisiksi osoitteiksi

Kuvassa ylimpänä oleva NEON-ydin auttaa prosessoriytimien toimintaa, kun käsitellään esimerkiksi mediaa. NEON laskee lähdön usealle tulovektorille tekemällä saman operaation jokaiselle tulovektorille samanaikaisesti (engl. SIMD) ja tallettamalla tuloksen vastaaviin lähtövektoreihin. Tämä on hyvä ratkaisu esimerkiksi kuvalle tai videolle, jossa tietomäärä (pikselit) on todella suuri. NEON:n ansiosta prosessorille jää aikaa hoitaa muuta laskentaa ja se sopii hyvin esimerkiksi FIR-suodattimen vasteen laskentaan tai nopean Fourier-muunnoksen (FFT) suoritukseen. NEON-ydin ei tue itsessään kaksoistarkkuuden lukuja ja sen vuoksi

kaksoistarkkuuden laskentaa varten sen yhteyteen on lisätty FPU-yksikkö (engl. Floating Point Unit).

APU:n ohella merkittävä oheislaite SCU (engl. Snoop Control Unit) toimii linkkinä prosessoriydinten ja välimuistien L1 ja L2 sekä oheismuistin (engl. On-Chip-Memory, OCM) välillä ja pitää huolen siitä, että välimuistit ovat keskenään johdonmukaisia. SCU myös hoitaa liikennettä ohjelmoitavaan logiikan ja suoritinpuolen välillä käyttämällä tähän ACP-porttia (engl. Acceleration Coherency Port), kuvassa 3 oikealla.

2.1.2 Suoritinpuolen liitynnät

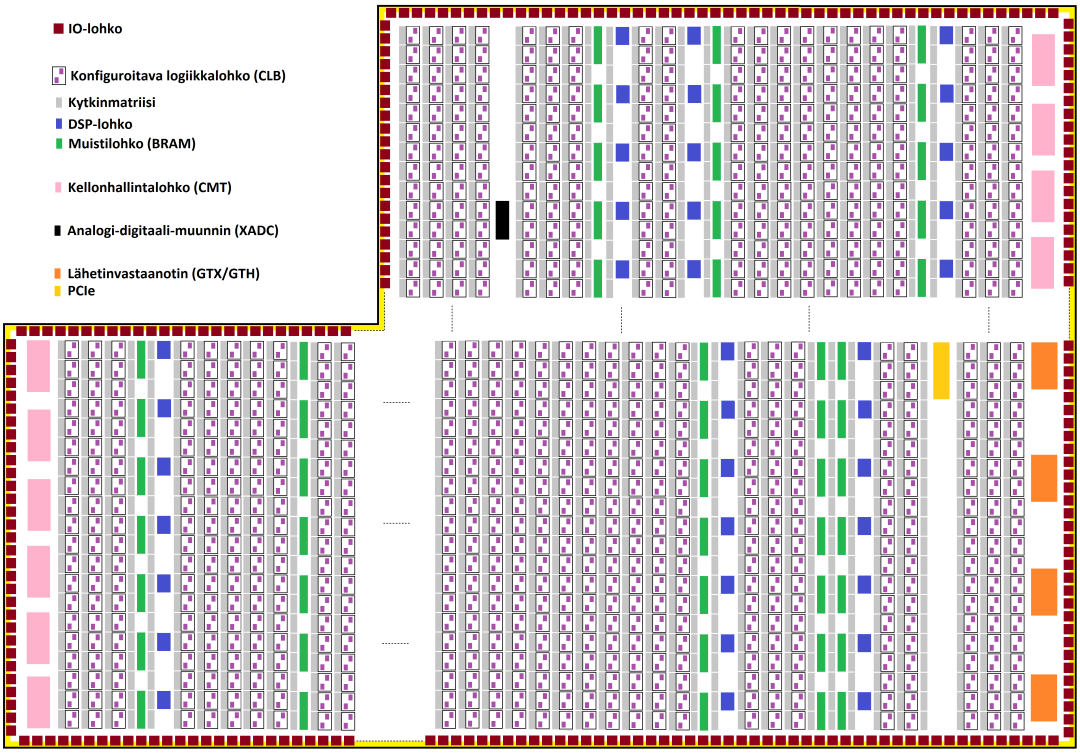
Suoritinpuoli keskustelee ulkomaailman ja ohjelmoitavan logiikan kanssa erilaisten liityntöjen (engl. interface) avulla. Zynqin ulkopuolisten laitteiden kommunikoinnista vastaa kuvan 2 vasemmassa reunassa näkyvät IO-liitynnät MIO ja EMIO. Näistä MIO (engl. Multiplexed Input/Output) on yhteydessä suoraan ulkomaailmaan, kun taas EMIO (engl. Extended MIO) kulkee ohjelmoitavan logiikan kautta ja jakaa IO-laitteita tämän kanssa. MIO tarjoaa 54-pinniä, jotka voidaan mapata halutulla tavalla ja EMIO tarjoaa mahdollisuuden laajentaa IO-pinnien määrää yli 54:n. Suorintinpuolen liityntöjä (AXI) FPGA-verkon puolelle ei tässä käydä läpi vaan kannattaa katsoa lähdeluettelon [3] s. 30-34 ja s. 353-364 (kpl. ”AXI Interfacing”).

2.2 OHJELMOITAVA LOGIIKKA JA SEN RESURSSIT

Ohjelmoitavan logiikan puoli ei suinkaan ole pelkkä kaksiulotteinen transistorimatriisi vaan verkolle on saatavilla perusresursseja, joita yhdistelemällä saadaan rakennettua suunnittelijan tarvitsema kokonaisuus. Resurssit on sijoiteltuna sinne sarakkeittain ja näitä ovat:

- konfiguroitava logiikkalohko (engl. Configurable Logic Block, *CLB*)
- muistilohko (engl. Block Random Access Memory, BRAM)
- digitaalisen signaaliprosessoinnin lohko (engl. Digital Signal Processing Block, DSP Block)
- yleiskäyttöinen Input/Output-lohko (IO-lohko)
- usean gigabitin lähetinvastaanotinlohko (engl. multi-Gigabit Transceiver, GT)
- kelloresurssit
- analogi-digitaalimuunnin (*XADC*) sekä
- integroitu PCIe -lohko

Resurssit sijaitsevat verkolla niin, että konfiguroitavat logiikkalohkot, dsp-lohkot sekä RAM-lohkot on hajautettu pitkin verkkoa ja mm. lähetinvastaanottimilla, IO-lohkoilla ja kelloresursseilla on muutama vakiintunut paikka jossain päin SoC:n reunaa. Seuraava kuva havainnollistaa resurssien sijoittumista Zynqillä.



Kuva 4. Zynqin ohjelmoitavan logiikan resurssit.

Kuvasta nähdään, että kaikki resurssit on sijoitettu ohjelmoitavalle logiikalle aina sarakkeittain. Hyvä on huomata, että peruslogiikka on sijoitettuna aina toistensa lähelle eli jos esim. digitaalista signaalinprosessointia (sinisellä) tarvitaan, on sen yhteyteen sijoitettuna myös muistia (vihreällä) ja konfiguroitavaa logiikkaa (violetti-valkoinen). Lähetinvastaanottimia, GT, (oranssilla) sijoitetaan yleensä neljän (nk. quad) rykelmäksi ja näiden läheisyyteen on integroituna PCIe-lohko (kultaisella), joka huolehtii luotettavasta sarjaväyläisestä tiedonsiirrosta Zynqiltä ulkomaailmaan. Jotta liikenne hoituisi myös koko verkon sisällä, on komponenttien väliin sijoitettu kytkinmatriisit (harmaalla), joita ohjelmoimalla saadaan yhdisteltyä resurssit keskenään. Lohko XADC

mahdollistaa mm. piirin itsenäisen lämpötilan ja sisäisten jännitteiden mittaamisen, joten säästytään muutamalta ulkoiselta anturilta.

Xilinx tarjoaa 7-sarjassaan neljää eri tuoteperhettä FPGA-verkon toteuttamiseen: Spartan-7, Artix-7, Kintex-7 ja Virtex-7. Ne on optimoitu eri käyttötarkoituksiin sopivaksi joten niiden suorituskyky, tehonkulutus, tuotannollinen hinta, kaistan määrä sekä käytettyjen resurssien lukumäärä eroaa toisistaan siirryttäessä tuoteperheestä toiseen.

Tuoteperheistä ”kevein” on Spartan-7, jonka resurssien määrää on pyritty karsimaan, jotta tehonkulutus jäisi mahdollisimman pieneksi. Artix-7:lla on paras suorituskyky mitä tulee tehonkulutukseen ja kaistan määrään. Kintex-7 puolestaan on kustannustehokas ja silti suorituskykyinen ja Virtex-7:lla on paras suorituskyky ja suurin valikoima eri resursseja. Zynq on optimoitu suhteellisen suorituskykyiseksi, joten se käyttää ainoastaan Artix-7 ja Kintex-7-perheitä. Taulukko 1 valottaa hieman resurssien määrää eri Zynq7000-sarjan laitteilla.

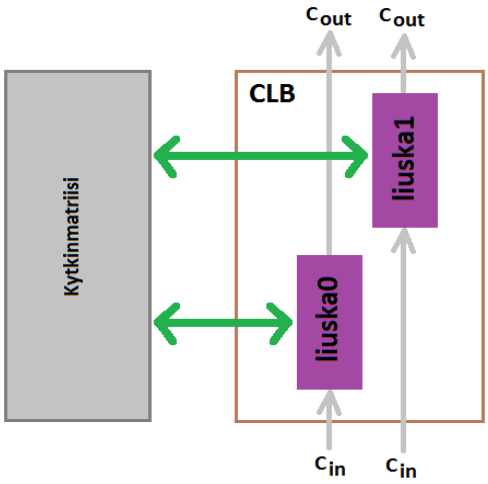
Taulukko 1. Luettelo Zynq7000-laitteiden ohjelmoitavan logiikan resurssien lukumäärästä (S=Spartan, A=Artix, K=Kintex, V=Virtex)

ZYNQ-LAITE	***CLB-liuskojen lkm***	***DSP-lohkojen lkm***	***36kB RAM-lohkojen lkm***
7S6	938	10	5
7S15	2000	20	10
7S25	3650	80	45
7S50	8150	120	75
7S75	12000	140	90
7S100	16000	160	120
7A12T	2000	40	20
7A15T	2600	45	25
7A25T	3650	80	45
7A35T	5200	90	50
7A50T	8150	120	75
7A75T	11800	180	105
7A100T	15850	240	135
7A200T	33650	740	365
7K70T	10250	240	135
7K160T	25350	600	325
7K325T	50950	840	445
7K355T	55650	1440	715
7K410T	63550	1540	795
7K420T	65150	1680	835
7K480T	74650	1920	955
7V585T	91050	1260	795
7V2000T	305400	2160	1292
7VX330T	51000	1220	750
7VX415T	64400	2160	880
7VX485T	75900	2800	1030
7VX550T	86600	2880	1180
7VX690T	108300	3600	1470
7VX980T	153000	3600	1500
7VX1140T	178000	3360	1880
7VH580T	90700	1680	940
7VH870T	136900	2520	1410

Seuraavissa kuudessa kappaleessa valotetaan tarkemmin mm. konfiguroitavan logiikkalohkon, digitaalisen signaalinprosessoinnin lohkon sekä muistilohkon toimintaa ja rakennetta. Myös joitain sisäisiä signaaleita käydään läpi, jotta toiminta selkeytyy helpommin. Yleisiä liityntöjä ulkomaailmaan (IO-lohkoja) avataan hieman mutta niiden rakenteisiin ei sen tarkemmin mennä. Muiden resurssien osalta riittänee pelkkä yleistuntemus ja mikä niiden rooli ohjelmoitavan logiikan puolella on.

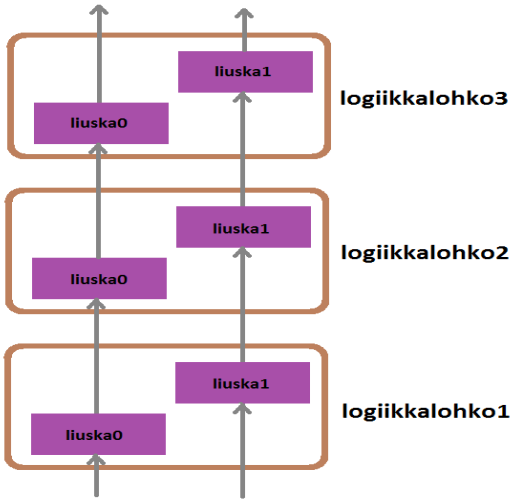
2.2.1 Konfiguroitava logiikkalohko

Konfiguroitava logiikkalohko (CLB) on ohjelmoitavan logiikan (FPGA-verkon) perinteisin resurssi, jolla voidaan toteuttaa sekä kombinaatio- että sekvenssilogiikkaa. Sen yleisenä käyttökohteena on aritmeettisten operaatioiden suoritus eli summaus-, vähennys-, kerto- ja jakolaskut. Se voidaan muuntaa myös muistiksi (RAM, ROM) ja sitä voidaan käyttää myös siirtorekistereihin ja multipleksereihin. CLB-lohkot on sijoitettu pitkin FPGA-verkkoa, ja täten niitä on helppo käyttää muiden resurssien, kuten dsp-lohkon, apuna. Kuvassa 5 on havainnollistettuna logiikkalohkon yksinkertaistettu rakenne.



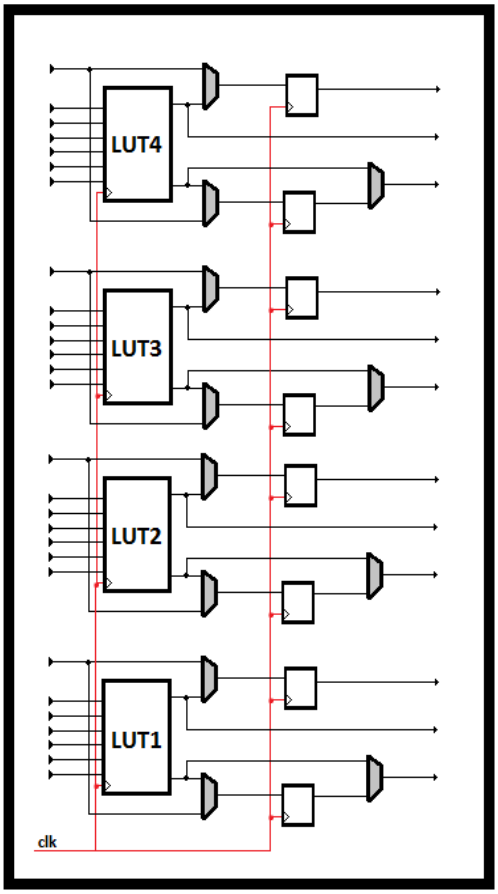
Kuva 5. Konfiguroitavan logiikkalohkon (CLB) yksinkertaistettu rakenne

Logiikkalohko koostuu kahdesta nk. liuskasta (engl. slice), jotka ovat kytkinmatriisin avulla yhteydessä muihin resursseihin sekä suoritinpuoleen. Liuskat nimetään kirjallisuudessa yleensä slice0:ksi ja slice1:ksi. Näiden välillä (eli slice0-slice1) ei ole keskinäistä yhteyttä, vaikka ne sijaitsevatkin samassa kokonaisuudessa. Ne voidaan kuitenkin ketjuttaa toisten logiikkasolujen samannimisten liuskojen kanssa liuskoilta löytyvää carry-logiikkaa apuna käyttäen. Tähän logiikkaan ei pääse käsiksi suoritinpuolelta eikä muun ohjelmoitavan logiikan kautta. Kuvassa 6 havainnollistetaan liuskojen välistä ketjutusta. Myös digitaalinen signaalinprosessoinnin lohkoja (dsp-lohkoja) voidaan ketjuttaa toisiinsa samankaltaisesti käyttämällä dsp-lohkojen omaa carry-logiikkaa. Dsp-lohkoista myöhemmin.



Kuva 6. Liuskojen ketjutus

Tässä vaiheessa mainittakoon, ettei liuskoja aina konfiguroida toteuttamaan sekä loogisia että muistillisia toimintoja, vaan osaa liuskoista yksinkertaistetaan alan säästämiseksi ja tällöin liuska toteuttaa vain loogisia toimintoja. Tällaista liuskaa kutsutaan nimellä SLICEL (L tullee sanasta logic). Monimutkaisempaa, muistitoimintoja sisältävää, liuskaa kutsutaan taas SLICEM:ksi (M=memory). Erittäin yksinkertainen malli liuskasta on esitetty kuvassa 7.

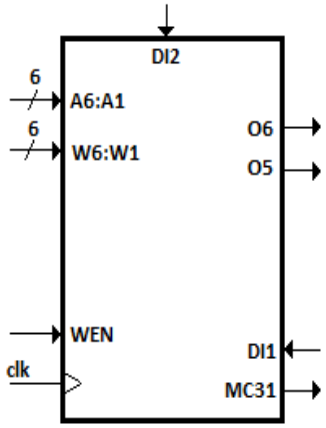


Kuva 7. Xilinx 7-sarjan logiikkaliuskan yksinkertaistettu rakenne.

Liuskalla on neljä pääosiota: neljä funktiogeneraattoria eli hakutaulukkoa (engl. Look-Up Table, LUT), kahdeksan muistielementtiä ja valintamultiplekserit sekä carry-logiikka. Carry-logiikka, hallintasignaalit sekä suurin osa valintamultipleksereistä on jätetty kuvasta pois yksinkertaistamisen vuoksi. Puhutaan seuraavaksi hieman tarkemmin liuskan kahdesta pääosiosta, eli funktiogeneraattoreista ja muistielementeistä.

2.2.1.1 Funktiogenerattorit eli hakutaulukot

Hakutaulukot (engl. Look-Up Table, LUT) ovat koko logiikkalohkon tärkeimpiä komponentteja, sillä ne kykenevät muodostamaan minkä tahansa loogisen operaation eli Booleanin toiminnon, kuten OR, AND, NAND, NOR, XNOR jne. Yksi LUT voi muodostaa tuloksen maksimissaan kuudesta muuttujasta eli puhutaan 6-tuloisesta hakutaulukosta. LUT on siitä kätevä, että se voidaan vaihtoehtoisesti konfiguroida kahdeksi maksimissaan viisituloiseksi funktiogeneraattoriksi, jolloin lähtöjä on kaksi. Tällöin yhteisestä kuudesta tulosta viisi jaetaan yhteisesti näihin kahteen pienempään LUT:iin. Mikäli tarvitaan 8-tuloinen LUT, saadaan sellainenkin muodostettua yhdistämällä liuskan kaikki neljä hakutaulukkoa yhdeksi käyttämällä tähän tarkoitukseen tarkoitettua kolmea valintamultiplekseriä. Käyttämällä apuna carry-logiikkaa, myös suurempituloiset hakutaulukot ovat mahdollisia; yhdistellään vain useamman logiikkalohkon liuskoja toisiinsa. Kuvassa 8 on esitettynä muistitoiminnallisen SLICEM-liuskan 6-tuloinen hakutaulukko.



Kuva 8. SLICEM-liuskan hakutaulukko.

Kuten kappaleen alussa mainittiin, voidaan logiikkalohkoa käyttää myös valintamultiplekserinä (engl. MUX), pienenä muistina (DRAM/ROM) tai siirtorekisterinä. Juuri hakutaulukot ovat ydin näiden rakentamiseen. Jotta näistä tärkeistä variaatioista tulisi selkä kuva, avataan hieman näitä kaikkia.

Muisteista ensimmäinen, eli DRAM on hajautettua (engl. Distributed) hajasaantimuistia (RAM). Se on nopeaa ja synkronista muistia, johon on mahdollista käyttää vain SLICEM-liuskan hakutaulukoita. Yhdestä hakutaulukosta voi rakentaa maksimissaan kuusituloisen, eli tällöin osoiteavaruuden maksimikoko on 64. Hakutaulukoita yhdistelemällä voidaan liuskalle muodostaa maksimissaan 8-bittinen osoiteavaruus eli tällöin maksimikoko on 256. Xilinxin 7- sarja tarjoaa useita eri vaihtoehtoja sekä osoiteavaruuksien koolle että dataporttien ja osoiteporttien määrälle. Taulukossa 2 on esitetty kaikki DRAM-tyypit.

Taulukko 2. Xilinx 7 -sarjan konfiguroitavan logiikkalohkon hakutaulukoista muodostetut hajautetun muistin (DRAM) perustyypit.

RAM-tyyppi	Tunniste	Hakutaulukoiden lkm
32 x 1S	RAM32X1S	1
32 x 1D	RAM32X1D	2
32 x 2Q	RAM32M	4
32 x 6SDP	RAM32M	4
64 x 1S	RAM64X1S	1
64 x 1D	RAM64X1D	2
64 x 1Q	RAM64M	4
64 x 3SDP	RAM64M	4
128 x 1S	RAM128X1S	2
128 x 1D	RAM128X1D	4
256 x 1S	RAM256X1S	4

RAM-tyypissä ensimmäinen arvo (32,64,128,256) on luonnollisesti liuskan osoiteavaruuden koko. Tyypin jälkimmäinen osio kertoo hakutaulukon dataporttien (esim. 1S, missä 1 = 1-bit tulo; 6SDP, missä 6 = 6-bit tulo) ja lukuosoiteporttien lukumäärän. Tulobittien ollessa 2 tai 6, hakutaulukoilla on yksi lisälähtö. RAM-tyypin merkinnässä kirjaimet määritellään luku/kirjoitusporteille seuraavasti:

- S = single port, eli luku- ja kirjoitusosoitteet jakavat saman osoitetulon
- D = dual port, ensimmäinen lukuosoite jaetaan ensimmäisen hakutaulukon lukuosoitetuloon ja kaikkien hakutaulukoiden kirjoitusosoitetuloon ja toinen lukuosoite jaetaan loppujen hakutaulukoiden lukuosoitetuloihin (poikkeuksena 128 x 1D, missä ensimmäinen lukuosoite jaetaan 1. ja 2. hakutaulukon lukuosoitetuloon ja toinen lukuosoite jaetaan 3. ja 4. hakutaulukon lukuosoitetuloon)
- Q = quad port, jokaiseen neljään hakutaulukkoon tulee oma lukuosoitteensa, kirjoitusosoite jokaiselle hakutaulukolle jaetaan ensimmäisen hakutaulukon lukuosoitteesta

SDP (engl. Simple Dual Port) eli yksinkertainen tuplaportti jättää yhdeltä hakutaulukolta datalähdön kokonaan pois.

Tarkempaa rakennetta eri DRAM-muistityypeistä kannata tässä yhteydessä käydä läpi. Lisätietoa löytyy lähdeluettelon [3] s. 23-34. Tässä vaiheessa riittää tietää, että DRAM on nopeaa RAM-muistia, joten se lisää prosessoinnin suorituskykyä. Sillä ei kuitenkaan kannata rakentaa suuria muisteja, sillä rakentaminen kuluttaa suhteettoman monta hakutaulukkoa ja ala kasvaa räjähdysmäisesti. Suurempiin muisteihin kannattaa sen sijaan käyttää hitaampia BRAM-resursseja, joista kappaleessa 4.3.

Toinen muistityyppi, johon hakutaulukot voidaan valjastaa, on ROM-muisti (engl. Read Only Memory) eli lukumuisti, johon ei voida kirjoittaa. Se on nopeaa 1-bittistä muistia ja voidaan muodostaa sekä SLICEL- että SLICEM-liuskoille. Muistien koot ja käytettyjen hakutaulukoiden määrä selviää seuraavasta taulukosta.

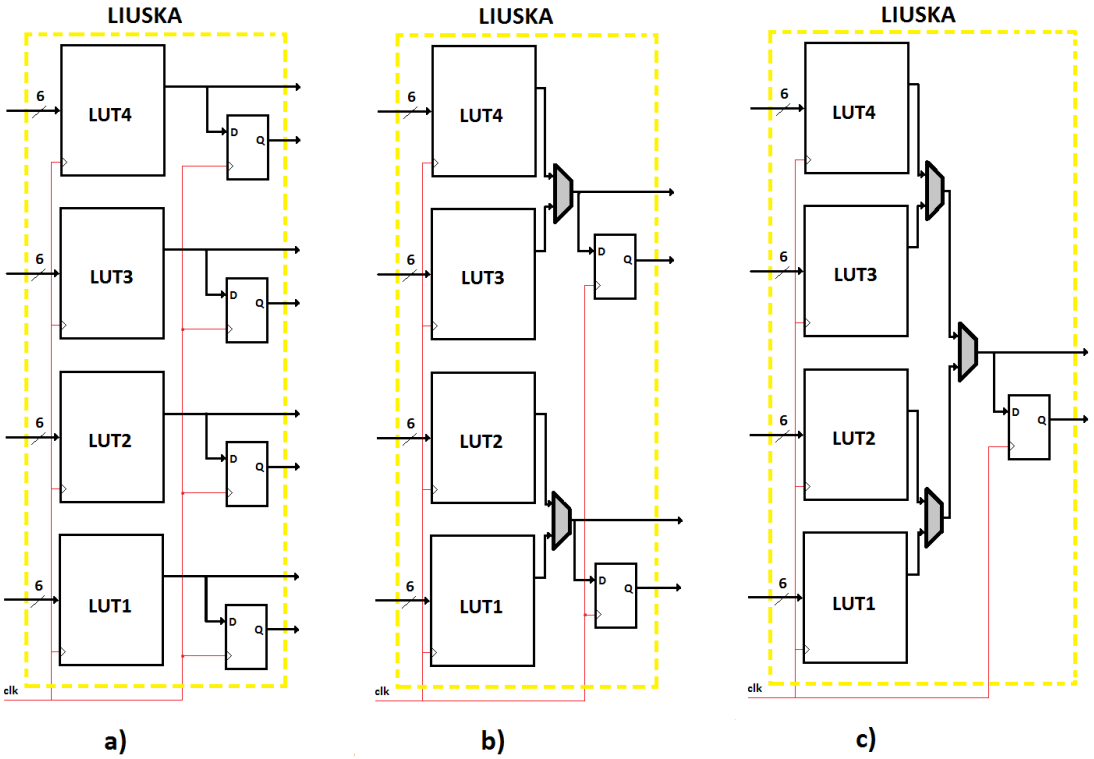
Taulukko 3. Xilinx 7 -sarjan konfiguroitavan logiikkalohkon hakutaulukoista muodostetut ROM-perustyypit

ROM	Hakutaulukoiden lkm
64 x 1	1
128 x 1	2
256 x 1	4

Kolmas kohde, johon hakutaulukoita voidaan käyttää, on **valintamultipleksereihin** (engl. MUX). Hakutaulukoita ja liuskalta löytyviä kolmea multiplekseriä yhdistelemällä voidaan rakentaa useampituloisia multipleksereitä. Vaihtoehtoja on kolme:

a) neljä kappaletta 4:1 MUX:ia (Kuva 9a)
b) kaksi paria 8:1 MUX (Kuva 9b) ja
c) yksi 16:1 MUX (Kuva 9c)

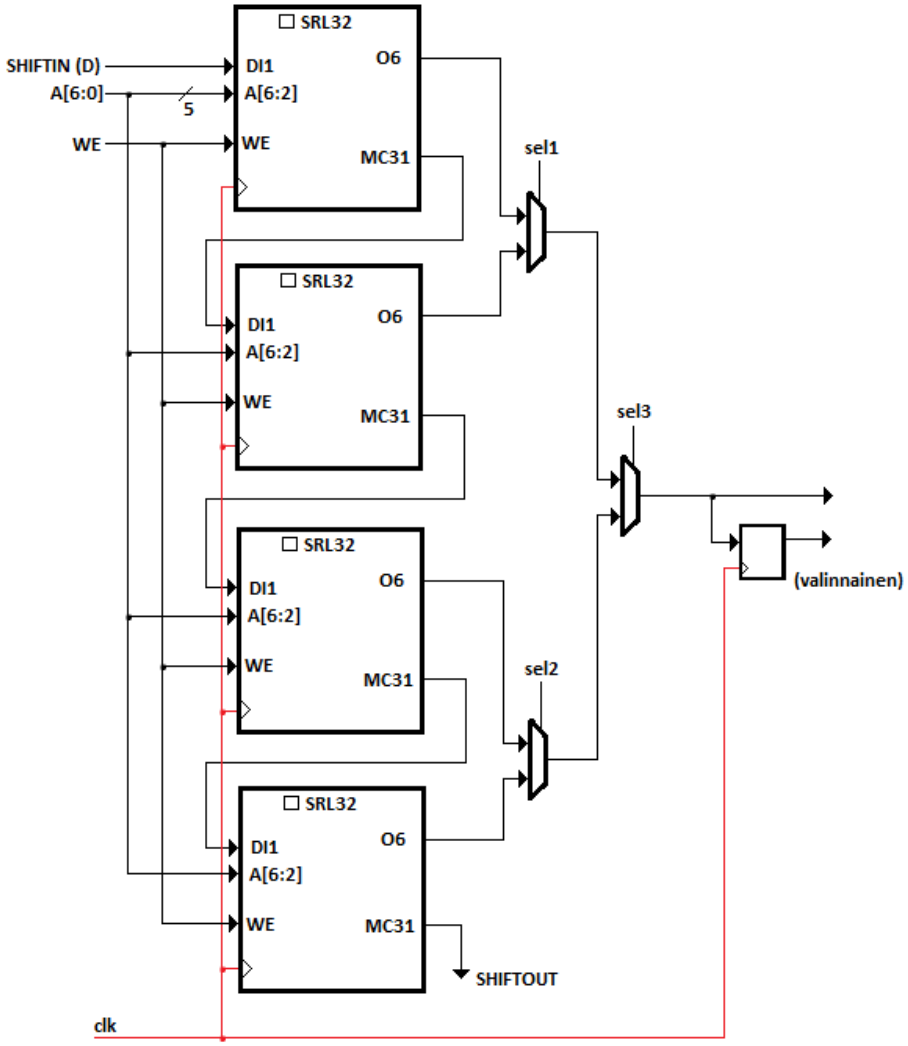
Merkille pantavaa on, että multiplekserin lähdöstä on saatavilla kaikissa tapauk-sissa sekä kombinaatio- että kellotettu lähtö. Tulossa näkyvä 6 bittiä tarkoittaa neljää datatuloa (DATA[3:0]) ja kahta valintasignaalia (SEL[1:0]). Multiplekserit voidaan rakentaa ainoastaan SLICEM-liuskalla.



Kuva 9. Xilinx 7-sarjan liuskan valjastaminen multiplekseriksi. a) 4kpl 4:1 MUX, b) 2kpl 8:1 MUX tai c) 1kpl 16:1 MUX.

Hakutaulukoiden neljäs mahdollinen käyttökohde ovat **siirtorekisterit** (engl. Shift Register, SR). Yhdestä hakutaulukosta voidaan muodostaa 32-bittinen siirtorekisteri, eli dataa voidaan tällöin viivästyttää 1-32 kellojaksoa ja tällainen on kätevää prosessien liukuhihnoituksessa. Mikäli halutaan viivästyttää dataa lisää, ketjutetaan vain hakutaulukoita toisiinsa niin, että hakutaulukon lähtöportti MC31 (kts. Kuva 8) ajetaan seuraavan hakutaulukon datatuloon DI1. Näin ollen liuskalle voi rakentaa 128-bittisen (4x32-bit) siirtorekisterin, joka viivästyttää dataa 128 kellojakson verran. On mahdollista rakentaa suurempiakin siirtorekistereitä liuskojen välillä, mutta suoraa linkkiä carry-

logiikan kautta ei ole, ja näin suunnittelija joutuu tyytymään vetoihin ohjelmoitavan logiikan kautta. Kuvassa 10 on esitetty 128-bittinen siirtorekisteri, jossa käytössä ovat liuskan (vain SLICEM) kaikki hakutaulukot.



Kuva 10. Neljästä hakutaulukosta rakennettu 128-bittinen siirtorekisteri.

Hakutaulukon osoitetulo A määrittää sen, mikä bitti ajetaan ulos hakutaulukosta. Näin saadaan määriteltyä se, miten paljon dataa viivästetään. Lopulta joku datan arvo ajetaan varsinaiseksi lähdöksi joko kiikun kautta (synkroninen lähtö) tai ilman (kombinaatiologiikkalähtö).

2.2.1.2 Muistielementit

Yksi liuska sisältää kahdeksan muistielementtiä, joista neljä toimii reuna-aktiivisina kiikkuina ja toiset neljä voidaan konfiguroida joko reuna-aktiivisiksi kiikuiksi tai tasoaktiivisiksi salvoiksi. Salpoja käytettäessä neljä normaalia kiikkua on pois käytöstä.

Kuvasta 7 nähdään, että liuskan ulostuloista on saatavilla sekä kellotettu (kiikun läpi kulkeva) että kombinaatiologiikkalähtö. Kellotettu lähtö mahdollistaa synkronoinnin muihin toimintoihin, mikä puolestaan mahdollistaa liukuhihnoituksen, jota tarvitaan aritmeettisten operaatioiden tehokkaassa suorituksessa.

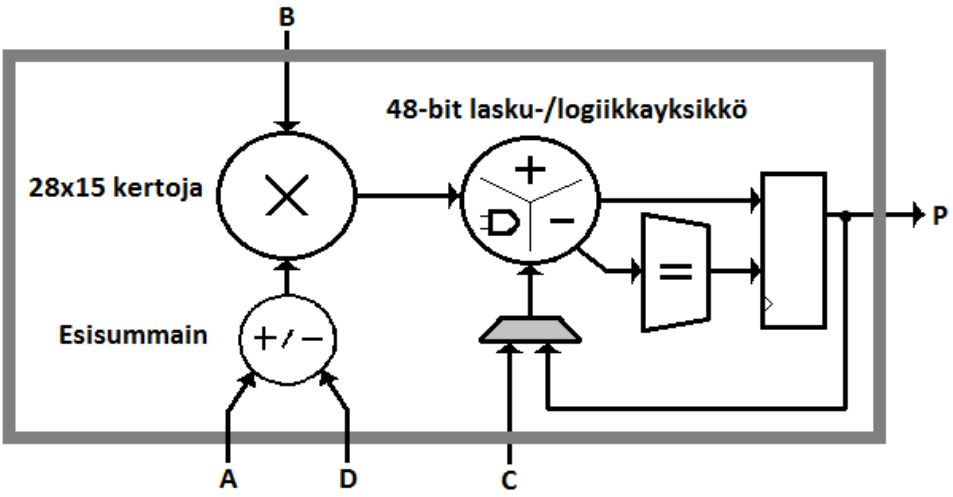
Tarkempaa rakennetta muistielementeistä ja hakutaulukoista hallintasignaaleineen ei tässä kannata alkaa esittämään, mutta halutessaan sellaista löytyy lähdeluettelossa [5] s. 15-23. Carry-logiikasta taas löytyy lisätietoa samaisesta manuaalista s. 43-44.

Vaikka logiikkalohkolla voidaankin käsitellä pitkien sananleveyksien aritmeettisia operaatioita, kannattaa sitä kuitenkin käyttää vain lyhyille sananleveyksille. Tämä siksi, että pitkät sananleveydet vaativat todella paljon liuskoja ja tämä kasvattaa liiaksi lopullisen komponentin alaa. Keskipitkille ja pitkille sananleveyksille käytetään yleensä dsp-lohkoa, josta seuraavassa.

2.2.2 Digitaalisen signaalinprosessoinnin lohko

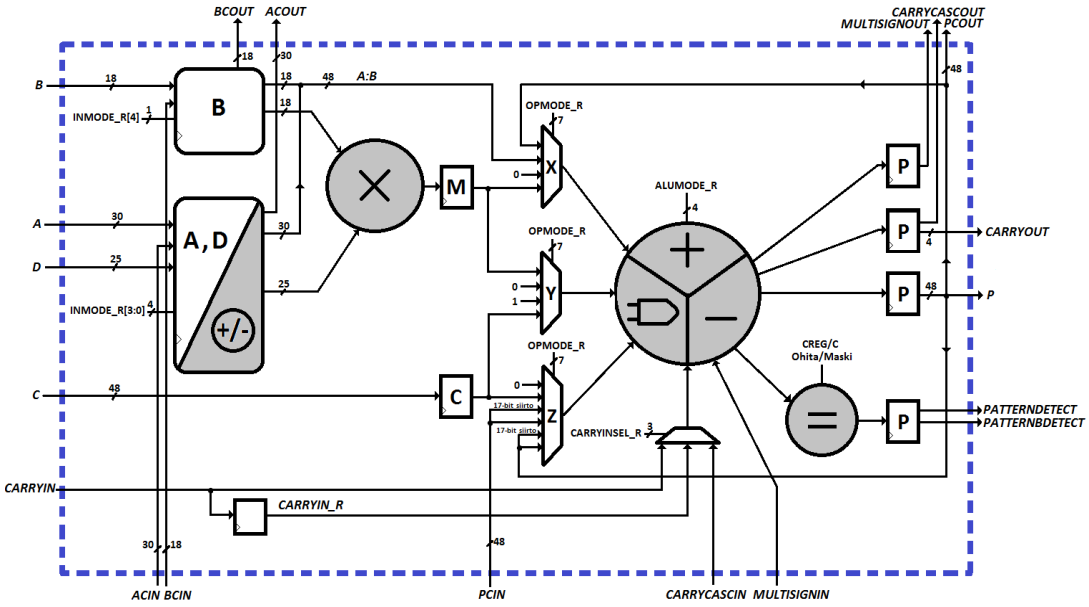
Tätä ohjelmoitavan logiikan resurssia, josta käytetään Xilinxin dokumentaatiossa nimeä DSP48E1, käytetään yleensä prosesseissa, joissa tarvitaan rinnakkaista laskentaa, esimerkiksi median käsittelyssä. Prosessoria pyritään käyttämään tehokkaasti hyväksi varsinaisen ohjelman suorituksessa ja mm. kuvien käsittelyä ajetaan dsp-lohkoille, joissa laskenta voidaan suorittaa samanaikaisesti itse pääprosessin rinnalla. Ilman dsp-lohkoja prosessori saattaa yksin tukehtua ja laskenta voi viivästyä niin, että muu prosessointi kärsii merkittävästi.

Myös sarakkeittain sijoitettuja DSP48E1-lohkoja voidaan ketjuttaa keskenään lohkon sisäisen carry-logiikan avulla. Näin saadaan toteutettua suurikertoimisiakin dsp-suodattimia ja mm. nopea Fourier-muunnoksen (engl. Fast Fourier Transform, FFT) laskeminen onnistuu järkevästi. Yhdellä lohkolla voidaan toteuttaa useita hitaampia operaatioita samanaikaisesti, joten koko piirin suorituskykyä saadaan helposti kasvatettua. Kuvassa 11 on esitetty erittäin yksinkertaistettu rakenne DSP48E1-lohkolle tuloineen (A,B,C ja D). Kuva on hieman harhaanjohtava siinä mielessä, että jokaisesta tulosta on saatavilla sekä kombinaatiologiikka- että kellotettu versio. Myös lähdölle P on saatavilla eo. vaihtoehdot.



Kuva 11. Xilinx 7 -sarjan DSP48E1-lohkon pelkistetty rakenne.

Lohkon tärkeimmät suorittavat osat ovat kaksituloinen 2-komplementti kertoja (datatulot vastaavasti 18- ja 25-bittisiä) sekä kolmituloinen 48-bit laskenta-/logiikkayksikkö. Lisäksi lohkolta löytyy esisummain sekä nk. pattern detect. Esisummaimessa voidaan summata tai vähentää 30-bittinen tulo A ja 25-bittinen tulo D. Tulo D on myös mahdollista ajaa valintamultiplekserillä suoraan kertojan tuloon, jolloin ei luonnollisesti suoriteta esisummausta. Pattern detect (”muodontunnistin”) on taas laskentayksikön lähdön vertailua varten: lähtöä verrataan johonkin tiedossa olevaan datajonoon. Toisin kuin CLB-lohko, on dsp-lohkon rakenne suhteellisen helppo hahmottaa, joten avataan hieman tarkemmin sen carry-logiikkaa, ohjaussignaaleita ja muuta rakennetta. Seuraavassa kuvassa on esitetty kattavampi rakenne.



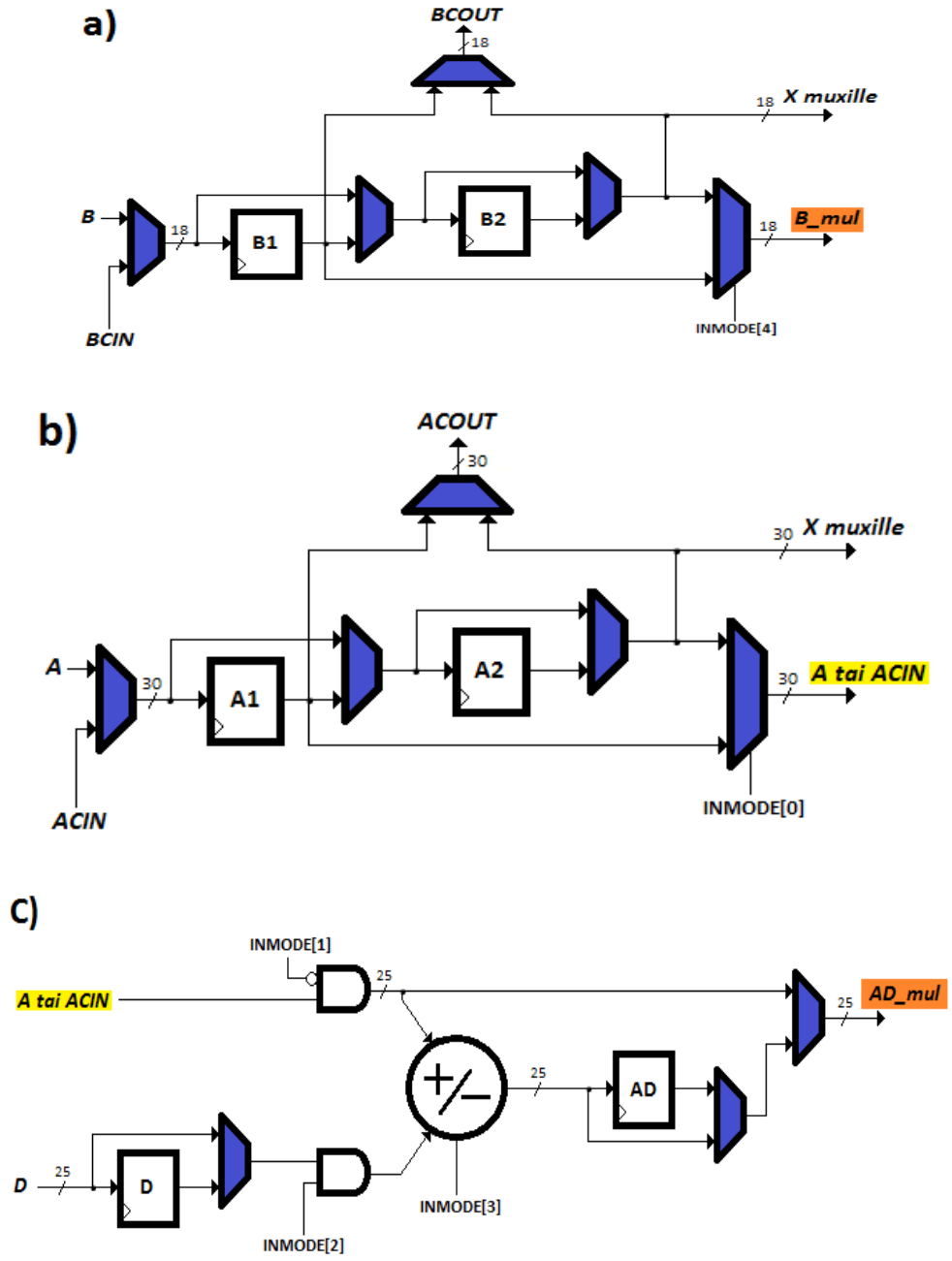
Kuva 12. Xilinx 7 -sarjan DSP48E1-lohkon kattavampi rakenne.

Kuvassa tumman harmaalla värillä on eroteltu vasemmalta lähtien: esisummain, kertoja, laskenta/logiikkayksikkö ja muodontunnistin. Käydään nämä kokonaisuudet läpi lähtien liikkeelle tulopuolen rekistereistä ja edetään kertojan ja valintamuxien kautta laskentayksikölle ja siitä lähtöön. Kuvassa vasemmalta oikealle kulkevat signaalit ovat yhteydessä ohjelmoitavaan logiikkaan kuten myös rekistereitä, multipleksereitä ja toiminnallisia lohkoja ohjaavat ohjaussignaalit. Alhaalta ylös kulkevat signaalit ovat dsp-lohkojen välisiä sisäisiä carry-signaaleita eikä niihin päästä käsiksi ohjelmoitavan logiikan puolelta. Kaikki ohjaussignaalit otetaan sisään rekisterin kautta (esimerkkinä CARRYIN), jotta mm. synkronointi onnistuisi. Nämä rekisterit on jätetty pois kuvan selkeyttämisen vuoksi. Ohjaussignaalin nimessä oleva R edustaa rekisteröityä tuloa.

2.2.2.1 Tulorekisterit (B, A/D) ja esisummain

Kuvassa 12 vasemmalla löytyy kaksi kaksoisrekisteriä B ja A, joiden tuloina ovat ohjelmoitavalta logiikalta tulevat signaalit B (18-bit) ja A (30-bit) sekä edelliseltä dsp-lohkolta tulevat BCIN (18-bit) ja ACIN (30-bit). A-rekisterin yhteydessä olevan D-rekisterin tulona on ohjelmoitavan logiikan signaali D (25-bit). Esisummaimella voidaan summata tai vähentää toisistaan rekisterin A lähtöä ja D-rekisterin lähtöä. Se on näppärä apu signaalien käsittelyyn ennen kertojaa.

Rekistereitä A, B, ja D sekä esisummainta ohjataan ohjaussignaalilla INMODE. Sen eniten merkitsevää bittiä (INMODE[4]) käytetään B-rekisterin ohjaukseen ja neljällä vähiten merkitsevällä bitillä (INMODE[3:0]) ohjataan A- ja D-rekisteriä sekä esisummainta. Jotta ohjaus olisi helpompi hahmottaa, katsastetaan rakenteita tarkemmin (Kuva 13).



Kuva 13. DSP48E1-lohkon a) rekisteri B, b) rekisteri A ja c) esisummaus.

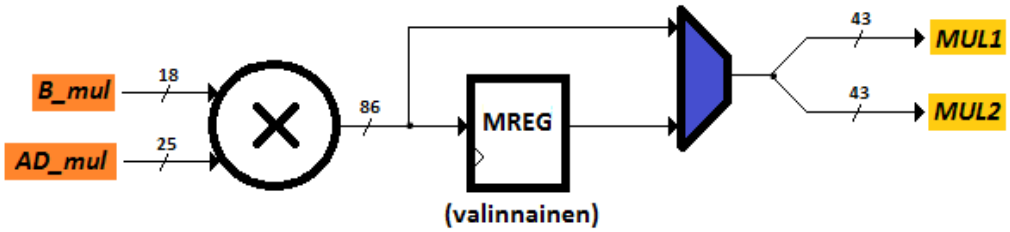
Rekisterikokonaisuuksien sisäiset multiplekserit päättävät, ajetaanko tulo suoraan sisään vai rekisterin kautta. Sama pätee, kun ohjataan A/ACIN ja B/BCIN seuraavalle dsp-lohkolle. Sisäisten muxien ohjaukseen käytetään omia ohjaussignaaleita, joita ei ole merkitty kuviin. Myös rekistereiden kellonsallintatulot ja reset-signaalit on jätetty selkeyttämisen vuoksi pois.

Kuvasta 13 c) nähdään, että kertojan alempaan tuloon voidaan viedä joko A, ACIN tai pelkkä D. Pelkän D:n saattaminen kertojan tuloon mahdollistetaan ohittamalla esisummain kokonaan. Kuvista a) ja b) nähdään, että rekistereiden A ja B yhdistetty lähtö viedään laskentayksikölle multiplekserin X kautta.

Tarkkasilmäinen on voinut jo havaita, että rekisterin A tulot ja lähdöt ovat 30-bittisiä ja esisummaukseen ja kertojalle menevät signaalit vain 25-bittisiä. Tämä johtuu siitä, että koko 30-bittinen tulo voidaan yhdistää 18-bittisen B:n kanssa niin, että laskentayksikölle saadaan vietyä yhteensä 48-bittinen signaali, eli rekisterit A ja B yhdessä väylässä. Tämä tulo voidaan merkitä A:B, missä A on signaalin ylimmät 30 bittiä ja B taas muodostaa alimmat 18 bittiä.

2.2.2.2 25x18 kertoja

Tämä kertoja on kaksituloinen nk. 2-komplementtikertoja, jonka tulot (25-bit ja 18-bit) ovat myös 2-komplementtimuotoisia. Se, mikä rekistereistä kytkeytyy tämän kertojan tuloiksi, määräytyy kohdan a perusteella. Tässä vaiheessa on hyvä selventää, että alkuperäisestä 30-bittisestä A (tai ACIN) tulosta on saatu 25-bittinen käyttämällä vain alinta 25-bittiä eli A[24:0] tai ACIN[24:0]. Kertoja tuottaa lähtöön kaksi kappaletta 43-bittistä (25+18=43) osatulosta, joista toinen (MUL1) johdetaan X-multiplekserille ja toinen (MUL2) taas Y-multiplekserille, joiden jälkeen ne summataan laskentayksikössä. Kertojan lähtö on täten näennäisesti 86-bittinen ja siinä voidaan haluttaessa käyttää liukuhihnarekisteriä (MREG).



Kuva 14. DSP48E1-lohkon 25x18 kertoja.

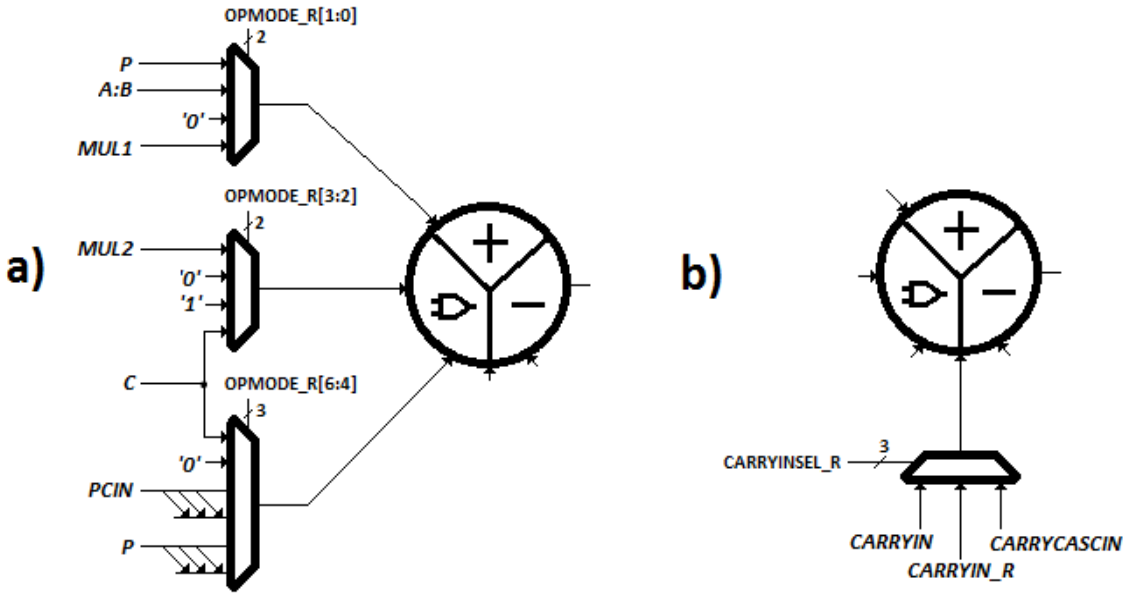
2.2.2.3 48-bittinen laskenta-/logiikkayksikkö

Laskentayksiköllä voidaan laskea yhteen tai vähentää multipleksereiltä X, Y, ja Z tulevia 48-bittisiä signaaleita sekä yksibittistä carry-signaalia. Näin ollen laskentayksikköä voidaan kutsua myös toisen asteen summaimeksi, esisummaimen ollessa ensimmäisen asteen summain. Pelkkää logiikkayksikköä käytettäessä käytössä on kolmesta tulosta kaksi, joille voidaan tehdä mikä tahansa looginen operaatio: AND, OR, NAND, NOR, NOT, XOR tai XNOR. Mikäli logiikkayksikköä käytetään, on kertoja oltava pois käytöstä, ja päin vastoin.

Laskentayksikön yksi tärkeä tehtävä on hoitaa kertojan aloittama työ loppuun. Toisin sanoen kertojan tuottamat osatulokset (MUL1 ja MUL2) summataan keskenään. Tällöin laskentayksikkö muuntuu nopeaksi 2-tuloiseksi summaimeksi. Summain suorittaa aina summauksen X+Y+CIN ja joko vähentää tai summaa tuloksen multiplekserin Z kanssa (1).

$$\text{2. asteen summain} = (Z +/- (X+Y+CIN)) \text{ tai } (-Z\ (X+Y+CIN)-1) \quad (1)$$

Kuvasta 12 havaitaan, että laskenta-/logiikkayksikkö saa yhteensä viisi tuloa. Näistä kolme tuloa on multipleksereiltä X, Y ja Z (Kuva 14a)). Neljäs tulo on nimeltään CIN (engl. Carry IN) eli carry-tulo, ja se saadaan multiplekseriltä, joka valitsee kolmesta eri carry-signaalista (Kuva 14b)). Viides tulo on puhtaasti edelliseltä DSP48E1-lohkolta tuleva MULTISIGNIN, joka ilmaisee edellisen lohkon kertojan tuloksen etumerkin. Tätä voidaan käyttää, kun laajennetaan 48-bittinen kertoja 96-bittiseksi. Katsastetaan multipleksereiden rakennetta hieman tarkemmin.



Kuva 15. Valintamultiplekserit a) X, Y, Z b) CIN.

Kuvassa 15 a) on esitetty multiplekserit X, Y ja Z, joiden ohjaamiseen käytetään rekisteröityä 7-bittistä signaalia OPMODE. Tämän kahta vähiten merkitsevää bittiä (eli OPMODE[1:0]) käytetään X-muxin ajamiseen, seuraavaa kahta (eli OPMODE[3:2]) Y-muxille ja eniten merkitseviä (eli OPMODE[6:4]) taas Z-muxiin. Tässä yhteydessä bittien kombinaatioilla ei ole väliä vaan pelkkä yleiskäsitys tulojen sijainnista riittänee.

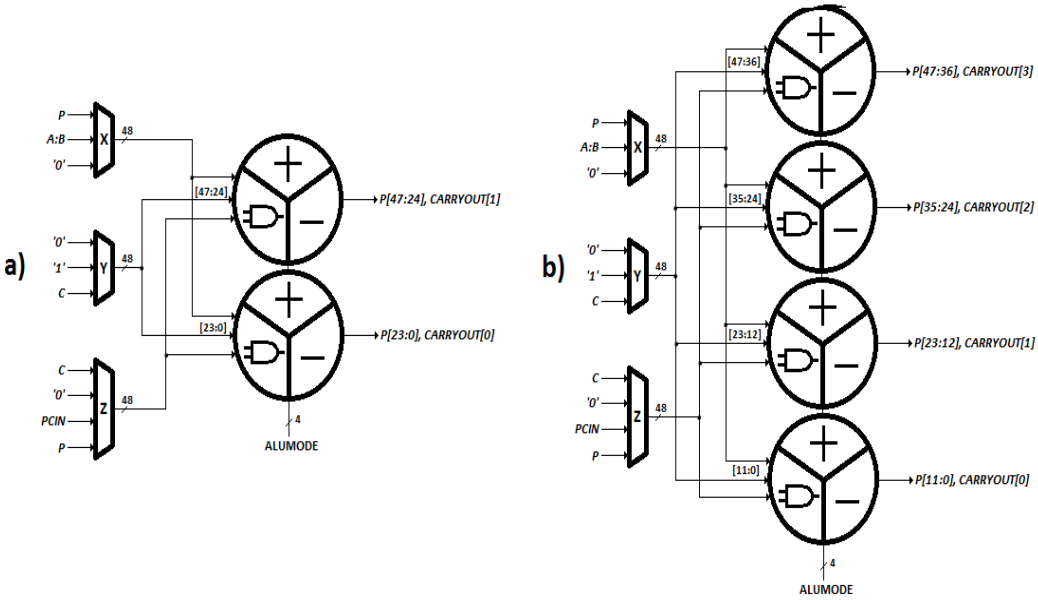
X-muxilta on mahdollista viedä laskentayksikölle kertojan osatulos MUL1, A- ja B-rekisterin yhdistelty 48-bittinen A:B tai sinne voidaan takaisinkytkeä laskenta-/logiikkayksikön lähtö P. Y-muxilta saadaan vastaavasti kertojan osatulos MUL2 tai rekisteröity tulo C. Tulo C voidaan johtaa

laskentayksikölle myös muxin Z kautta. Kaikkien laskentayksikön kolmen tulon bitit voidaan myös vaihtoehtoisesti nollata (tulo '0'). Y-muxilla on lisäksi mahdollista asettaa kaikki bitit loogiseen '1' (tulo '1').

Z-muxin alaosassa näkyvät lisäksi oudot laskentayksiköiden tulot P ja PCIN (edellinen liuska). Niitä käytetään, kun halutaan siirtää tulon bittejä 17-bitin verran oikealle. Tämä on tarpeellista suurempien kertojien rakentamisessa liuskoja yhdistelemällä. Ei mennä tähän sen tarkemmin.

Kuvan 15 b) multiplekserin tehtävä on toimittaa laskentayksikölle carry-signaali dsp-lohkojen ketjutusta varten. Se valitsee, otetaanko sisään FPGA-verkolta tuleva carry-signaali (CARRYIN) vai edelliseltä lohkolta tuleva carry-signaali (CARRYCASCIN).

Laskenta-/logiikkayksikkö tukee nk. SIMD-toimintoa (engl. Single Instruction, Multiple Data), joka sallii tulodatalle tehtävän samat toiminnot. Tämä tarkoittaa, että jokaiselle datan osalle tehdään sama toiminto, esimerkiksi kuvankäsittelyssä tuttu kirkkauden säätö jokaiselle pikselille. Laskenta-yksiköllä SIMD-toiminto voidaan toteuttaa pilkkomalla kolmituloinen (tai kaksituloinen) 48-bittinen aritmeettinen operaatio usealle lohkolle samanaikaisesti tapahtuvaksi. Siinä yksikkö jaetaan joko kahteen 24-bittiseen tai neljään 12-bittiseen yksikköön (Kuva 16), joista jokainen suorittaa saman toiminnon, esimerkiksi kolmi- tai kaksituloisen summauksen. Sama jako voidaan tehdä myös logiikkayksikölle.



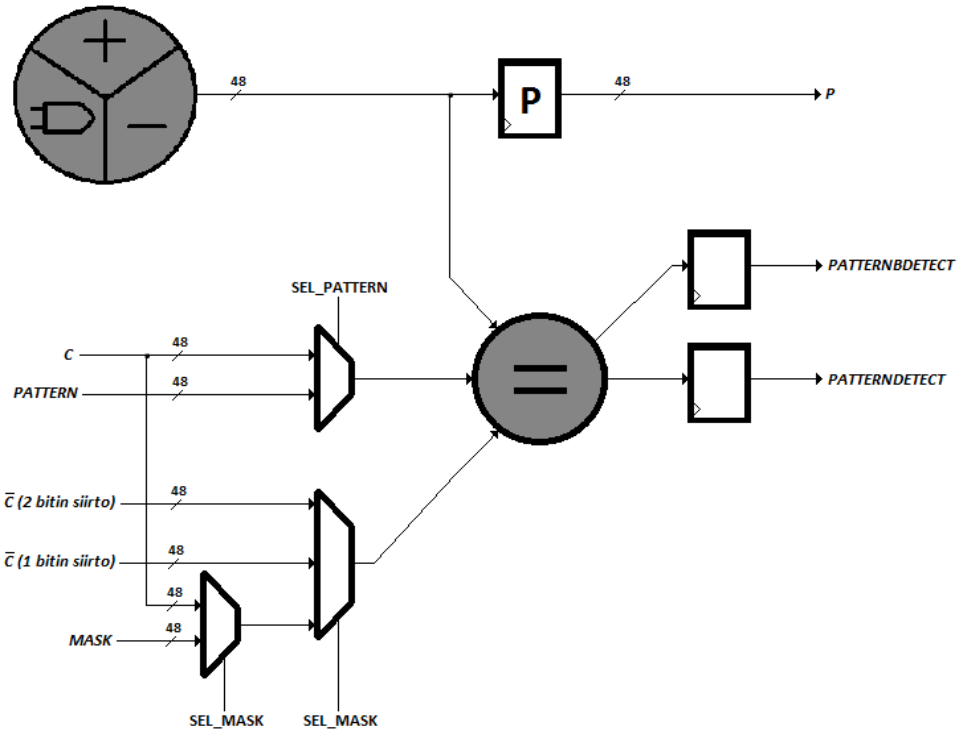
Kuva 16. Laskenta-/logiikkayksikön SIMD-toiminto: a) 2x 24-bit tai b) 4x 12-bit.

2.2.2.4 Pattern detect

Kuten jo aiemmin mainittiin, tämä yksikkö (”muodontunnistin”) saa tulonsa laskenta-/logiikkayksiköltä. Tässä vaiheessa kannattaa muistaa, ettei tulo ole sama kuin rekisterin P lähtö. Tunnistinyksikkö antaa lähtönsä kylläkin samalla kellon reunalla P:n kanssa.

Laskenta-/logiikkayksiköltä saadun tulon lisäksi tunnistinyksiköllä on myös tulo, johon vertaillaan (joko rekisteri C tai haluttu mallidata (PATTERN)) sekä valinnainen maski. Näitä kaikki on havainnollistettu kuvassa 17. Vertailu voidaan tehdä parista eri lähteestä ja muutamaa erilaista maskia apuna käyttäen.

Tämä yksikkö on aika helppo ymmärtää, sillä se vertailee vain laskenta- ja logiikkayksikön tuloa joko malliin tai maskiin ja mikäli datat ovat samoja, asetetaan bitti PATTERNDETECT tai PATTERNBDETECT loogiseen '1'. Näitä signaaleita voidaan sitten käyttää ylivuodon ja alivuodon tunnistukseen. Mikäli esimerkiksi havaitaan, että vertaillut tulot ovat samoja ja arvo sanotaan vaikka 0001111... niin ylivuoto tapahtuisi, mikäli arvo on 0010000... tai yli. Alivuoto taas tapahtuu mikäli arvo on 1110000... tai alle. PATTERNDETECT / PATTERNBDETECT asetetaan tällöin loogiseen '0' ja ajetaan erilliseen logiikkaan sisäisen signaalin (ei merkitty kuvaan) kanssa.

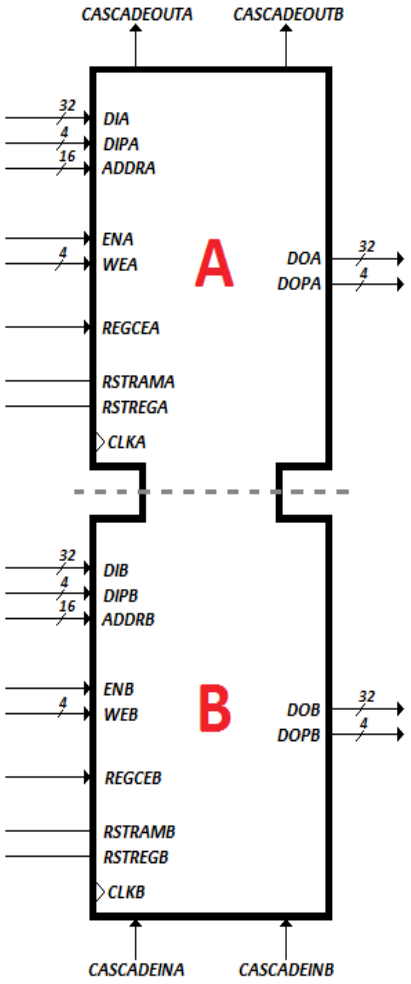


Kuva 17. Muodontunnistus valinnaisella maskilla.

Tunnistinyksiköllä voidaan todentaa datojen samankaltaisuus tilanteissa kuten A:B==C, AxB==C, A:B (C) ==0 ja monia muita. Yksikkö ei rajoitu pelkästään muodontunnistuksiin vaan sitä voidaan käyttää mm. 48-bittisen laskurin automaattiseen resetointiin ja se sopii hyvin myös lukujen pyöristyksiin.

2.2.3 Muistilohko

Muistilohko, josta käytetään tarkempaa nimitystä RAMB36E1, on tärkeä lisä ohjelmoitavan logiikan resursseihin. Siihen voidaan tallettaa maksimissaan 36kB dataa ja se voidaan myös jakaa kahdeksi itsenäiseksi 18kB muistiyksiköksi (RAMB18E1). RAMB36E1 ja RAMB18E1 ovat nk. todellisia kaksiportteja (engl. True Dual-Port, TDP), eli muistiin on pääsy kahden itsenäisen portin kautta. Näillä porteilla (A ja B) muistiin/muistista voidaan kirjoittaa/lukea samanaikaisesti muistiosoitteiden ADDRA ja ADDRB osoittamasta paikasta tai toista porttia voidaan käyttää kirjoittamiseen ja toista vastaavasti lukemiseen. Ao. kuvassa on havainnollistettu muistilohkoista suurempi (RAMB36E1).

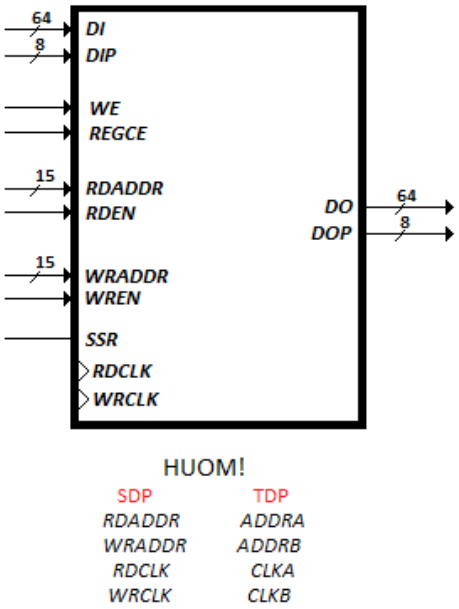


Kuva 18. Xilinx 7 -sarjan muistiresurssi RAMB36E1.

Lohkon kaikki sisääntulevat signaalit ohjataan rekistereihin synkronoinnin helpottamiseksi ja dataulostulot voidaan ajaa joko rekistereihin tai salpoihin. Salpatoiminto on normaalisti valittuna mutta rekisteröityä optiota suositellaan käytettäväksi mm. liukuhihnoituksen mahdollistamiseksi. Sekä luku- että kirjoitusoperaatiot ovat synkronisia ja niitä kellotetaan eri kelloilla (CLKA ja CLKB).

Kuvasta 18 voidaan havaita sisään- ja ulostulevan datan olevan 36-bittistä. Se on jaettu 32-bittiseen dataan (engl. Data In, DI – Data Out, DO) ja lisänä on neljä bittiä (engl. Data In Parity, DIP – Data Out Parity, DOP), joita voidaan käyttää pariteetti-, virhe- tai jopa lisädatabitteinä. Vastaavasti RAMB18E1-lohkolla data on 18-bittistä, josta datatulo (DI) muodostaa 16 bittiä ja pariteettibittejä on kaksi. Ohjaustuloihin ei kannata sen enempää paneutua.

Molempaa muistilohkoa voidaan käyttää myös nk. yksinkertaisena kaksiporttina (engl. Simple Dual-Port, SDP), jolloin käyttöön saadaan tuplamäärä dataa (36→72 bittiä / 18→36 bittiä). Tällöin portti A toimii lukuporttina ja portti B vastaavasti kirjoitusporttina. Yksinkertainen kaksiportti mahdollistaa edelleen samanaikaisen muistiin kirjoittamisen ja sieltä lukemisen. Kuvassa 19 on havainnollistettu yksinkertainen kaksiportti RAMB36E1.



Kuva 19. Yksinkertainen kaksiportti RAMB36E1.

SDP-muistia on mahdollista jakaa edelleen pienempiin lohkoihin. On myös mahdollista rakentaa suurempi 64kB lohko yhdistämällä suoraan kaksi vierekkäistä samalla sarakkeella olevaa RAMB36E1 toisiinsa, ja tällöin lohkon sisällä olevan logiikan ansiosta ulkopuolista logiikkaa CLB-puolelta ei tarvita lainkaan. Suurempien muistien rakentamisessa täytyy kuitenkin turvautua ulkopuolisen logiikan käyttöön. Seuraavaan taulukkoon on listattuna mahdolliset muistien koot ja lukumäärät.

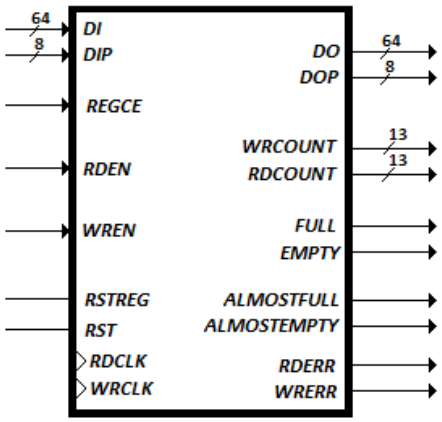
Taulukko 4. Varsinaisten muistien koot ja määrät RAMB36E1 ja RAMB18E1 -lohkoilla (vain SDP-moodi)

RAMB36E1	RAMB18E1
64K x 1	-
32K x 1	-
16K x 2	-
8K x 4	16K x 1
4K x 9	8K x 2
2K x 18	4K x 4
1K x 36	2K x 9
512 x 72	512 x 36

2.2.3.1 FIFO

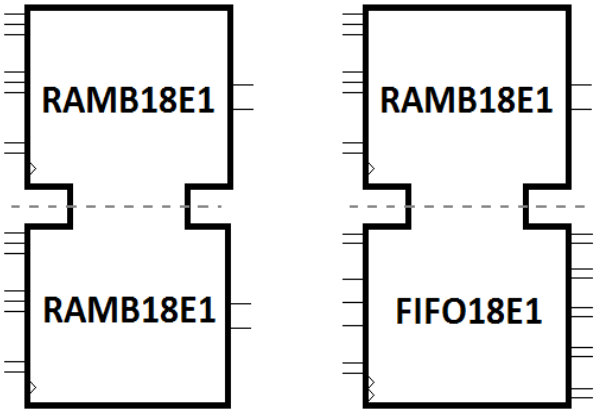
RAM-muistilohkon sisältämä ylimääräinen logiikka mahdollistaa lohkon valjastamisen suoraan joko 36kB tai kahdeksi 18kB FIFO-muistiksi (engl. First In First Out). Tällöin muistiosoitetuloja (ADDRA tai ADDRB) ei luonnollisesti ole, sillä FIFO-muistin laskurit (ja ohjaussignaalit) pitävät huolen, että muistin seuraavaan paikkaan siirrytään muistipaikan ollessa täysi tai muistipaikan lukemisen tullessa päätökseen. FIFO-muisteja kannattaa käyttää mm. mediadatan puskuroinnissa ja siirryttäessä nopeasta kellodomainista hitaampaan.

Kuvasta 20 nähdään, että FIFO sisältää melkomoisen määrän ohjaussignaaleita, joilla ilmoitetaan kirjoitus- ja lukutilanteen edistyminen sisäiselle ohjauslogiikalle. Signaaleita on yhteensä kahdeksan, joiden merkitys käy melko helposti selville jo niiden nimistä. Kuten RAM-muisteja, voidaan FIFO-muistejakin ketjuttaa. Mikäli halutaan suurempaa muistikokonaisuutta, sarjoitetaan FIFO36-muisteja, ja mikäli halutaan enemmän databittejä, laitetaan FIFO36-muisteja rinnakkain. Molemmat lisäykset vaativat ulkopuolista logiikkaa (CLB-resurssit).



Kuva 20. RAMB36E1 valjastettuna FIFO36-muistiksi.

Vaikka eo. tekstistä saakin kuvan, että yksi RAMB36E1-lohko voidaan valjastaa joko kahdeksi TDP-RAM-muistiksi, kahdeksi SDP-RAM-muistiksi tai kahdeksi FIFO18-muistiksi niin näin ei kuitenkaan ole. On mahdollista yhdistää RAMB18 ja FIFO18 samalle 36kB muistille (Kuva 21).



Kuva 21. Mahdolliset variaatiot RAMB36E1-lohkolla.

Pienimpään muistilohkoon eli 512x64 RAM yhteyteen voidaan lisätä 72-bittinen Hamming-virheenkorjaus (engl. Hamming Error Correction Code, Hamming ECC), jossa datan osuus on 64 bittiä ja virheenkorjaukseen käytetään loppuja kahdeksaa pariteettibittiä. Sillä on mahdollista havaita yhden ja kahden bitin virheet, mutta vain yhden bitin virheet voidaan korjata. Mikäli haluaa lisätietoa virheenkorjauksesta, kannattaa katsoa [7] s. 71 → .

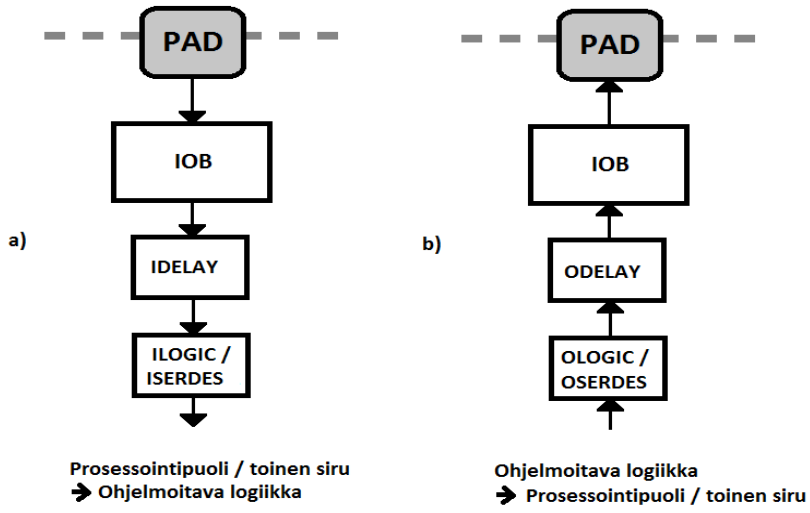
2.2.4 Yleiskäyttöiset IO-resurssit

Input-Output-lohko (engl. Input-Output Blocks, IOB) toimii linkkinä ohjelmoitavan logiikan ja ulkomaailman välillä. IO-lohko on Xilinxin tuoteperheiden tapauksessa tutummin SelectIO.

SelectIO-resurssit jaetaan karkeasti kahteen pääkategoriaan: korkean suorituskyvyn (engl. High Performance, HP) ja laajan skaalan (engl. High Range, HR) IO-lohkoon. Näistä ensimmäistä (HP-lohko) käytetään, kun tarvitaan nopea yhteys esimerkiksi muistiin tai toiselle sirulle. Laajan skaalan (HR) lohko taas antaa mahdollisuuden käyttää jopa 3.3V käyttöjännitettä. Nämä lohkot sopivat hitaammille ja yleisemmille IO-standardeille.

SelectIO koostuu puskurointi-osasta ja lisälogiikasta. Näistä ensimmäinen hoitaa linjojen sovituksen haluttuun impedanssiin (yleensä 50 ohmia). Sovituksilla yritetään eliminoida mahdollisia signaaleiden heijastuksia ja soimista. Logiikkaosalla on mahdollista viivästyttää signaaleita ja muuttaa sarjadata rinnakkaisdataksi ja päinvastoin. SelectIO:n korkean suorituskyvyn IO-lohko (HP-lohko) on esitetty kuvassa 22, missä vasemmalla on esitettynä FPGA-verkolle tulevan datan kulkua ja oikealla taas ulkomaailmaan menevän datan kulkua.

Ennen kuin signaalit saadaan FPGA-verkon resurssien käyttöön, ajetaan ne ensin kolmelle erillisen lohkon läpi (a). Puskurointilohkossa (IOB) signaaleille tehdään mm. sovitukset, jonka jälkeen signaaleita on mahdollista viivästyttää IDELAY-logiikan avulla. ILOGIC-lohkon tehtävänä on rekisteröidä tulevat signaalit ennen niiden saattamista FPGA-resurssien käyttöön. ISERDES-lohkossa voidaan sarjadatasta tehdä rinnakkaisdataa. Kuvassa b signaalit johdetaan ulkomaailmaan ja luonnollisesti FPGA-resursseilta tuleville signaaleille tehdään päinvastaiset operaatiot (vrt. kuva a). Laajan skaalan lohko (HR-lohko) eroaa HP-lohkon rakenteesta vain siinä mielessä, että signaaleita ei pystytä viivästämään (OLOGIC puuttuu) ennen niiden saattamista ulkomaailman käyttöön.



Kuva 22. SelectIO-resurssin rakenne (HP-lohko), a) ulkomaailmasta ohjelmoitavalle logiikalle (FPGA-verkolle) ja b) ohjelmoitavalta logiikalta (FPGA-verkolta) ulkomaailmaan.

2.2.4.1 DCI

Normaalisti suunnittelijan on lisättävä joku referenssivastus linjan yhteyteen, jotta se saadaan sovitettua signaalipolkuun. Tämä referenssivastus on yleensä melko kookas eikä se välttämättä mahdu sovitettavan linjan välittömään läheisyyteen tai halutun pinnin viereen. Varsinkin korkeaa suorituskykyä omaavissa prosesseissa vastuksen lisääminen ei ole enää mielekästä, sillä vastus lisää komponenttilukumäärää ja väistämättä pinta-alaa ja viiveitä. Juuri tätä ongelmaa varten Xilinx on kehittänyt näppärän tavan sovittaa lähtöajureiden lähtöjä ja vastaanotinten tuloja siirtolinjoihin: digitaalisesti kontrolloitu impedanssi (engl. Digitally Controlled Impedance, DCI). Siinä valitaan joku haluttu referenssivastus (esimerkiksi 50 tai 100 ohmia), joka on jo lohkon sisällä valmiina. Näin ylimääräistä vastusta ei lohkon ulkopuolelle tarvita. DCI on mahdollista matalien jännitteiden (<=1.8V) HP-lohkoissa.

IO-resurssien rakennetta ei kannata sen syvemmin käydä läpi. Lisätietoa voi hakea [8] s. 15 → (sovitukset) ja s. 106 → (ILOGIC, OLOGIC, ISERDER/OSERDES). Riittää, kun tietää piirinlaajuisen sovituksen olevan mahdollista (esim. DCI) ja mm. referenssijännitteen voi luoda IO-lohkon sisälle. Myös slew-raten ohjelmallinen säätö onnistuu niin HR- kuin HP-lohkoissa.

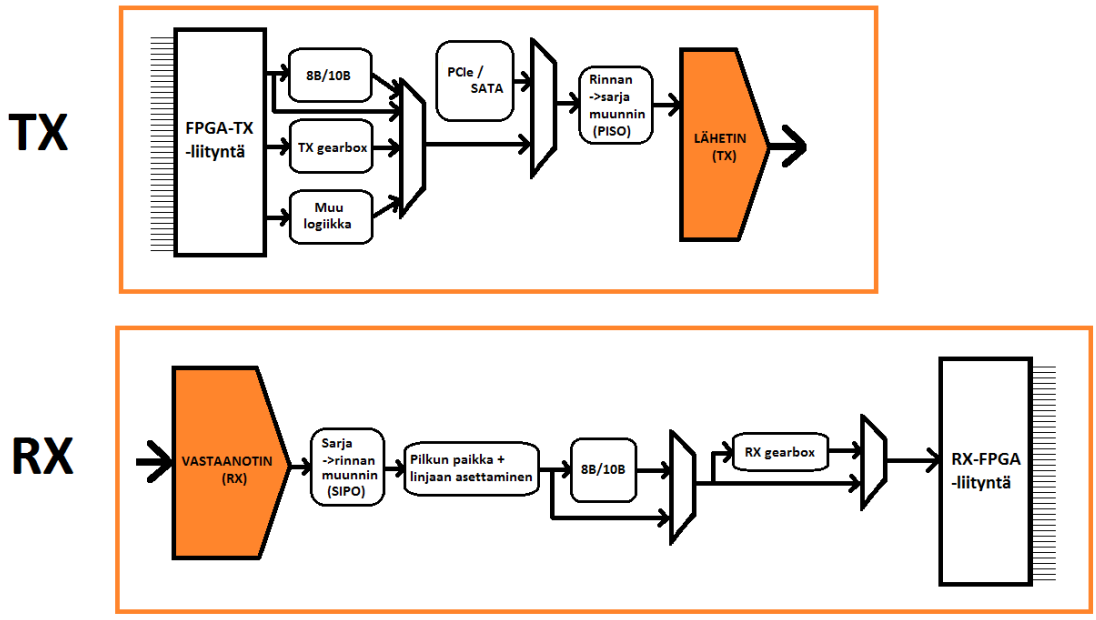
2.2.5 GTX/GTH-lähetinvastaanotin

Jotta dataa saadaan siirrettyä Zynq-piiriltä nopeasti ulkomaailmaan tai ulkomaailmasta Zynqille, tarvitaan tätä varten luotettava lähetin-vastaanotin (engl. transceiver). Piirien alkuaikoina pidemmille matkoille suosittiin sarjaliikenteistä rakennetta, jota käytettiin datan siirtoihin mm. tulostimille (COM-portti). Piirien monimutkaistuessa alettiin suosimaan rinnakkaisväyläisiä ratkaisuja (esim. PCI, engl. Peripheral Component Interconnect).

Rinnakkaisväyläiset ratkaisut siirtävät mm. videodataa tehokkaasti, mutta datan määrä on kasvattanut pinnien lukumäärää jo niin suureksi, ettei käytettävissä olevien fyysisten pinnien määrä enää yksinkertaisesti riitä. Sarjamuotoinen siirtotapa sitä vastoin ei tarvitse läheskään samaa pinnimäärää, vaikka protokolla vaatiikin enemmän virtapinnejä. Se kykenee siirtämään dataa yli 10GB/s nopeudella. Tällaisia usean gigabitin sekuntivauhdilla toimivia lähetinvastaanottimia kutsutaan yleisesti nimellä Multi-Gigabit SERDES (engl. SERializer DESerializer). Siinä piirillä käsitelty rinnakkaisdata muunnetaan sarjadataksi, joka lähetetään kanavaan ja vastaanottopiirin vastaanotin muuntaa datan takaisin rinnakkaiseen muotoon.

Zynqin FPGA-verkolle voidaan näppärästi alustaa tällainen sarjamuotoinen SERDES, josta käytetään nimitystä GT (engl. multi-Gigabit Transceiver). Zynqillä käytettävien GTX/GTH-lähentinvastaanottimilla lähetys/vastaanottonopeudet ovat 500MB-12.1GB/s (GTX) ja 500MB-13,5GB/s (GTH). Vastaanottimilla GTP siirtonopeus on hieman alhaisempi ja GTZ-mallilla yli 20GB/s. Sisäänrakennetun 8B/10B enkoodaus/dekoodaus-rakenteen avulla on mahdollista enkoodata 8-bit datasta 10-bit symboleita, jotka siirtyvät piiriltä piirille luotettavasti, koska nämä symbolit ovat uniikkeja ja vastaanotinpäässä huomataan virheet dekoodauksen jälkeen.

Kuvassa 23 on esitetty GTX/GTH lähetinvastaanottimen rakenne yksinkertaistettuna. Lähetinpuolta kutsutaan yleisesti nimellä TX ja vastaanotinta vastaavasti RX. 8B/10B koodaus-dekoodauslogiikka on mahdollista ohittaa ja lähettää data sellaisenaan kanavaan ja sama pätee vastaanottopuoleen. Kuvassa mystinen rakenne TX gearbox / RX gearbox tarjoaa mahdollisuuden 64B/66B koodaus/dekoodaus-laajennukseen. TX-puolella ”Muu logiikka” on esimerkiksi esimerkkijonon luominen / liukuhihnoituksen ohjaus. RX-puolella on oltava logiikka pilkun paikan tunnistamiselle sekä datan saattamiselle linjaan. Kellodomainit sekä muu logiikka on jätetty merkitsemättä kuvaan selkeyttämisen vuoksi.



Kuva 23. GTX/GTH-lähetinvastaanottimen yksinkertaistettu rakenne.

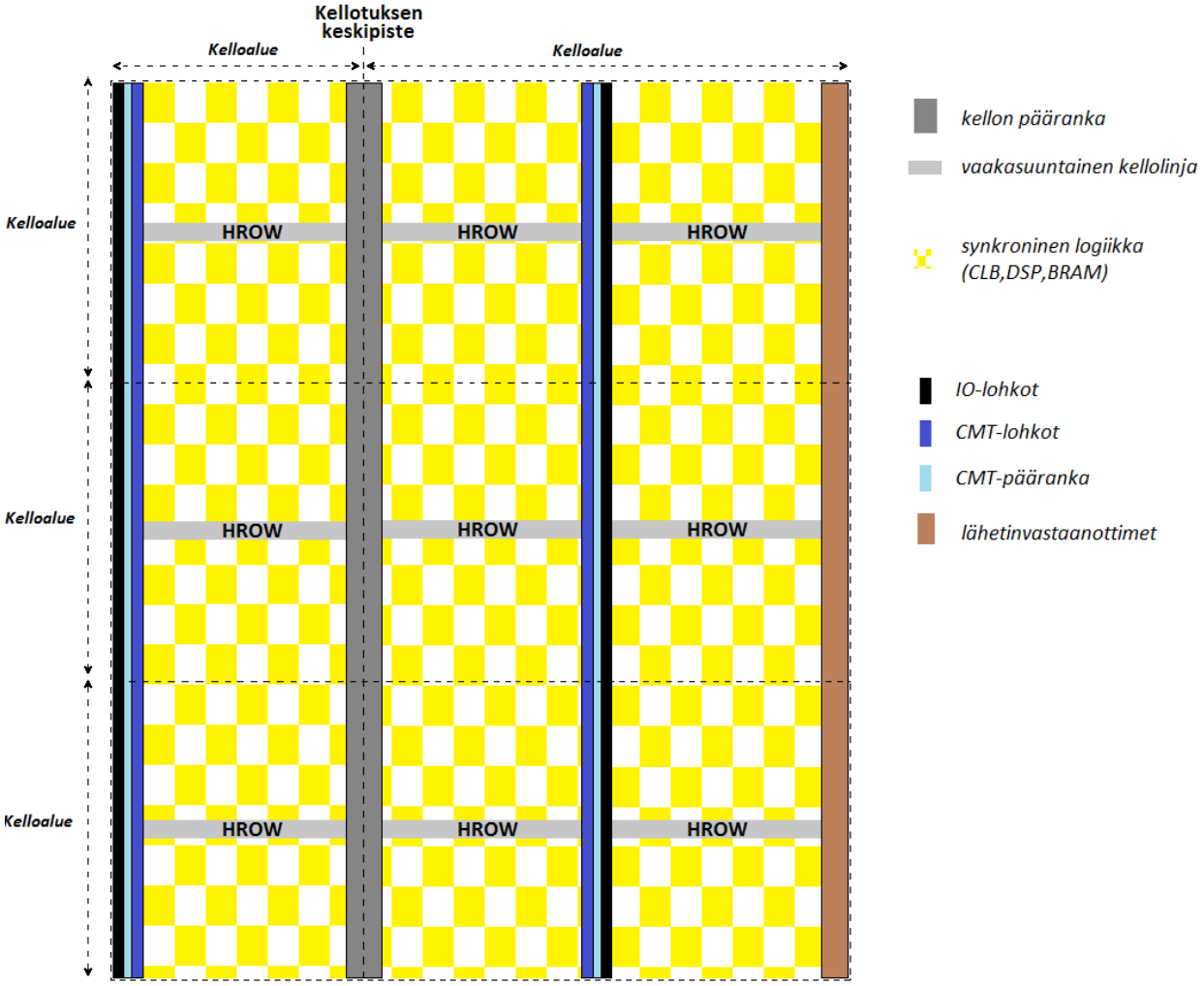
GT-lähetinvastaanottimen hyviä puolia ovat sen suorituskyky ja luotettavuus verrattuna rinnakkaiseen toteutukseen. Pinnien pienempi määrä vähentää myös alan tarvetta, joka vaikuttaa suoraan tehonkulutukseen, varmennusaikoihin ja näin ollen piirin hinta pienenee. Huonoihin puoliin lukeutuu signaalin eheyteen liittyvät analogiset lisäsimuloinnit, jotka hidastavat luonnollisesti varmennusprosessia ja näin ollen tuovat lisäkustannuksia ja viivyttävät tuotteen saamista markkinoille.

Yleensä GTX/GTH-lohkoja sijoiteltaessa FPGA-verkolle luodaan kaikki lähetinvastaanotin-lohkot yhdelle piirin sivulle. Niitä sijoitetaan normaalisti vähintään neljä yhdeksi kokonaisuudeksi, ja yhdellä GTX/GTH-lohkolla on oma vaihelukittu silmukkansa (CPLL) ja suurempia siirtonopeuksia varten nämä neljä lähetinvastaanotinlohkoa jakavat yhden monimutkaisemman vaihelukitun silmukan (QPLL).

2.2.6 Kelloresurssit

Zynqin ohjelmoitava logiikka (FPGA-verkko) on jaettu 1-24 kelloalueeseen, joiden sisään kaikki synkroniset komponentit eli CLB, DSP, BRAM, IO, GTX/GTH, CMT) on sijoitettuna.

Korkean tason kuvaus kelloalueista on havainnollistettuna kuvassa 24. Kuvasta nähdään, että yhden kelloalueen sisällä on aina keskellä vaakasuunnassa kulkeva kellon pääväylä (engl. Horizontal Row, HROW), ja alue levittäytyy tämän keskilinjan ylä- ja alapuolelle 25 logiikkasolun korkeuden verran. Koko FPGA-verkon kellon pääranka (engl. Clock Backbone) kulkee kellotuskeskipisteen kautta pystysuunnassa koko piirin yli. Tämän kautta otetaan FPGA-verkolla käytettävät kellosignaalit sisään kellopinnien ja globaalien kellopuskuroiden (BUFG) kautta. Näiden kellopuskureiden kautta voidaan ajaa kelloa mihin tahansa kellotuspisteeseen piirillä.



Kuva 24. Korkean tason esitys Xilinx 7 -sarjan kelloarkkitehtuurista.

Kuvassa vasemmalla on havainnollistettu IO-lohkojen sijoittumista, joiden vieressä kulkee kellonhallintalohkojen (engl. Clock Management Tile, CMT) sarake. Kuten nähdään, nämäkin resurssit on sijoitettu sarakkeittain. CMT-pääranka (engl. Backbone) mahdollistaa pystysuuntaisten CMT-lohkojen ketjuttamisen. Oikealle on sijoitettuna nopeiden sarjaväyläisten lähetinvastaanotinten sarake.

Globaaleiden kellopuskureiden lisäksi kelloresursseihin lukeutuvat IO-sarakkeella sijaitsevat puskurit BUFIO ja BUFR (R=regional), joiden kautta kellolla on pääsy IO-lohkoille ja synkroniselle logiikalle, jota on havainnollistettu kuvassa keltaisella. Kaikkien kellopuskureiden

päätehtävä on piirin synkronointi, mutta ne voivat toimia myös kellonsallintatuloina (CE) tai multipleksereinä, joilla voidaan valita eritaajuinen kello tiettyyn sovellukseen sopivaksi ja näin säästää tehossa. Myös vaihtaminen viallisesta kellodomainista pois onnistuu bufferin avulla.

Kellonhallintalohko koostuu MMCM (engl. Mixed-Mode Clock Manager) ja vaihelukitusta silmukasta (PLL). Lohkolla voidaan muodostaa eritaajuisia kelloja jakamalla kahta referenssikellosignaalia MMCM:ltä ja PLL:lta löytyvällä jakajalla. MMCM on vaihelukittua silmukkaan monipuolisempi, sillä se sisältää mm. edistyneemmän murtolukujakajan.

Vaikka Xilinx 7 -sarjan kelloalueiden instantointi onkin melko suoraviivainen prosessi kattavien ja johdonmukaisten ohjelmien avulla, pitää suunnittelijan kiinnittää huomiota mm. useiden kelloalueiden yli rakentuviin clb-, dsp- ja bram-lohkoihin. Näiden laajempien kokonaisuuksien kellottaminen on nimittäin järkevää tehdä käyttämällä suoraan io- (BUFIO) ja alueellisia (BUFR) puskureita.

Xilinx 7-sarjan kelloresurssit ovat joustavia, sillä niillä täytetään niin yksinkertaiset kuin kompleksisetkin kellotusvaatimukset. Kellonhallintalohkot tarjoavat hyvän kellotaajuussynteesin ja taajuushuojunnan suodatusta ja kellopuidensa ansiosta kelloviiveet saadaan minimoitua.

Lisätietoa kelloalueen rakenteesta löytyy [10] s.17-20 ja tarkempaa tietoa kelloresursseista samaisesta manuaalista s. 29 lähtien.

2.2.7 Analogia-digitaalimuunnin

Xilinx tarjoaa 7-sarjassaan analogia-digitaalimuuntimen, XADC, joka voi muuntaa differentiaalisen analogiatulon 12-bittiseksi digitaaliseksi signaaliksi miljoonan näytteen sekuntivauhdilla (1MSPS). Lisäksi XADC voi käyttää osan tai kaikki digitaalisista 16 differentiaaliparistaan analogisina lisätuloina.

Sisäänrakennettujen antureidensa avulla XADC:lla voidaan mitata mm. lämpöä ja jännitelähteiden jännitteitä (mm. BRAM). Mitattavat jännitealueet ovat tällöin 0-1V (unipolar-moodi) ja (-0.5)-0.5V (bipolar-moodi) ja pienimmäksi jännite-askeleeksi tulee 244uV (1V / 4096) 12-bitin tarkkuudella. XADC käyttää differentiaalista näytteistystä, ja tämän ansiosta yhteismuotoista kohinaa ei juurikaan ilmene.

2.2.8 PCI Express – yleiskäyttöinen I/O-ydin

Vaikkakin PCI (engl. Peripheral Component Interconnect) on alkujaan rinnakkaisprotokolla, on PCI Express (PCIe) sarjaväyläinen ratkaisu. Xilinx 7 -sarja tarjoaa lähetinvastaanotinten yhteyteen tällaisen integroitavan ytimen, joka on todella joustava, skaalattava ja luotettava, yleiskäyttöinen ydin, jolla voidaan saavuttaa jopa 5GB/s siirtonopeus. Siinä on integroituna myös 8B/10B-enkoodaus/dekoodaus, joka vähentää merkittävästi vastaanottopään virheitä. Se käyttää GTX/GTP lähetinvastaanotinrakenteita.

3. YHTEENVETO JA POHDINTA

FPGA-SoC-piirien tavoin Zynq sopii erinomaisesti alustaksi sovelluksiin, jotka vaativat joustavuutta, ja se on melkein ASIC-piirin luokkaa suorituskyvyssä mm. videonprosessoinnissa. Tämä on mahdollista logiikkaresurssien CLB, DSP48E1 ja RAMB36E1 ansiosta, joita voidaan helposti yhdistää toisiinsa kytkinmatriisien avulla. Lisäksi nopean carry-logiikan avulla samanalaisia lohkoja voidaan ketjuttaa ja näin ollen saadaan suurempia dsp-lohkoja (mm. suodattimet) , multipleksereitä (DSP48E1/CLB), muisteja, siirtorekistereitä (CLB) jne. Taulukossa 5 on havainnollistettu esimerkkinä kevyimmän (7A12T) ja tehokkaimman (7K480T) Zynq-laitteen CLB-liuskojen maksimimuisti- ja -siirtorekisterikapasiteettia, sekä vertailun vuoksi vielä clb-liuskojen, dsp-lohkojen ja bram-muistin maksimimäärät.

Taulukko 5. Zynq-laitteiden 7A12T ja 7K480T CLB-, DSP48E1- sekä RAMB36E1-resurssien määrät ja CLB-resurssien maksimikapasiteetit hajautetuttuna muistina (DRAM) tai siirtorekistereinä (SR)

Zynq-laite	CLB lkm	DSP48E1 lkm	RAMB36E1 [MB]	DRAM [kB]	SR [kB]
7A12T	2000	40	0,2	171	86
7K480T	74650	1920	34,38	6788	3394

Nopean sarjamuotoisen tiedonsiirtonsa (>>10GB/s) ja tuettavien siirtoprotokolliensa (mm. SATA, PCIe) ansiosta Zynq soveltuu erinomaisesti juuri multimedian käsittelyyn. Lisäksi Zynq:llä olevat IO-resurssit vähentävät linjojen sovituksiin tarvittavien komponenttien määrää, joka puolestaan vähentää alaa ja tehonkulutusta. Lisäksi io-pinniä voidaan käyttää joko nopeaan tiedonsiirtoon tai pinnille voidaan ajaa suuremman jännitteen signaaleita, joten sitä voidaan käyttää joko nopeassa tai hitaammassa tiedonsiirrossa. Selkeän ja joustavan kellonjakelun (kelloranka, bufferit, CMT) ja kattavan kellopuurakenteensa ansiosta kellon etenemisviiveet ja jitteri pienenevät koko piirin mittakaavassa. Tämä puolestaan lisää luotettavuutta merkittävästi.

Zynqille saatavilla lukuisia valmiita IP-lohkoja. joihin lukeutuu mm. kätevä apuprosessori MicroBlaze ja lukuisat muut Xilinxin omat ja kolmansilta osapuolilta saatavat pehmeät ytimet. Zynqin tulevaisuus näyttää hyvältä, sillä jopa 2GHz kellotaajuudella toimiva ARM Cortex Dual core -prosessori yhdessä 8MB välimuistinsa ja kevyen Linux-käyttöjärjestelmän kanssa prosessoi multimediaa tehokkaasti ja tuskinpa 4k-videon käsittelyynkään ihan heti tukehtuu.

Raskaalle kuvan- ja videonkäsittelylle on ominaista, että sananleveys saattaa olla joissain tilanteessa suhteettoman suuri. Ohjelmoitavan logiikan resurssit ovat todella käteviä myös tässä, että käytettävän datan sananleveyttä voidaan muuttaa CLB-lohkon avulla, mikäli vaikka 32/64-bittinen sananleveys ei enää riitä. Kuvan- ja videonkäsittelyssä todella tehokkaat (suurikertoimiset) FIR-suodattimet on mahdollista toteuttaa dsp-lohkoja ketjuttamalla. Joustavuutta lisää vielä dsp-lohkojen sisäiset rakenteet, joilla voidaan valita, käytetäänkö lohkoja niin, että suorituskyky on paras vai vähennetäänkö tehontarvetta suorituskyvyn kärsiessä. Tällaista joustavuutta eivät ASIC-piirit tarjoa, joten Zynq on tässä mielessä ylivoimainen toteutusalusta.

Zynqille on olemassa lukuisia mielenkiintoisia sovellusratkaisuja, joista esimerkkinä uusimpien joukossa mm. liikennemerkkien tunnistusta [13] tekoälyn neuroverkon kiihdytystä [14] sekä kuvapohjaista kasvojentunnistusta [15].

4. LÄHDELUETTELO

[1] What is a System on Chip (SoC)? URL: https://anysilicon.com/what-is-a-system-on-chip-soc/

[2] Rajsuman R. (2006) System-on-a-chip Design And Test. Artech House.

[3] Crockett L. H., Elliot R. A., Enderwitz M. A., Stewart R. W. (2014) The Zynq Book: Embedded Processing with the ARM Cortex-A9 on the Xilinx Zynq-7000 All Programmable SoC, First Edition, Strathclyde Academic Media.

[4] Jayaraman R. (2001) (When) Will FPGAs Kill ASICs. URL: https://www.doc.ic.ac.uk/~wl/teachlocal/arch/killasic.pdf

[5] Xilinx Inc (2016) 7 Series FPGAs Configurable Logic Block, User Guide, UG474 (v1.8). URL: https://www.xilinx.com/support/documentation/user_guides/ug474_7Series_CLB.pdf

[6]Xilinx Inc (2018) 7 Series DSP48E1 Slice, User Guide, UG479 (v1.10). URL: https://www.xilinx.com/support/documentation/user_guides/ug479_7Series_DSP48E1.pdf

[7] Xilinx Inc (2019) 7 Series FPGAs Memory Resources, User Guide, UG473 (v1.14). URL: https://www.xilinx.com/support/documentation/user_guides/ug473_7Series_Memory_Resources.pdf

[8] Xilinx Inc (2018) 7 Series FPGAs SelectIO Resources, User Guide, UG471 (v1.10). URL: https://www.xilinx.com/support/documentation/user_guides/ug471_7Series_SelectIO.pdf

[9] Xilinx Inc (2018) 7 Series FPGAs GTX/GTH Transceivers, User Guide, UG476 (v1.12.1). URL: https://www.xilinx.com/support/documentation/user_guides/ug476_7Series_Transceivers.pdf

[10] Xilinx Inc (2018) 7 Series FPGAs Clocking Resources, User Guide, UG472 (v1.14). URL: https://www.xilinx.com/support/documentation/user_guides/ug472_7Series_Clocking.pdf

[11] Xilinx Inc (2018) 7 Series FPGAs and Zynq-7000 SoC XADC Dual 12-Bit 1 MSPS Analog-to-Digital Converter, User Guide, UG480 (v1.10.1). URL: https://www.xilinx.com/support/documentation/user_guides/ug480_7Series_XADC.pdf

[12] Xilinx Inc (2020) 7 Series FPGAs Integrated Block for PCI Express v3.3, LogiCORE IP Product Guide, PG054. URL: https://www.xilinx.com/support/documentation/ip_documentation/pcie_7x/v3_3/pg054-7series-pcie.pdf

[13] Liu J., Zhang Z., Zheng L., Wen Y., Bin F., Tang L. (2021) Traffic Sign Recognition Based on ZYNQ. In: 2021 9th International Symposium on Next Genration Electronics (ISNE), Changsha. URL: https://ieeexplore.ieee.org/document/9493630.

[14] Lee H. S., Jeon J. W. (2021) Accelerating Deep Neural Networks Using FPGAs and ZYNQ. In: 2021 IEEE Region 10 Symposium (TENSYMP), Jeju. URL: https://ieeexplore.ieee.org/document/9550853.

[15] Živković M., Herceg M., Pjevalice N., Subotić M. (2021) Face Detection in Images Using an FPGA. In: 2021 Zooming Innovation in Consumer Technologies Conference (ZINC), Novi Sad. URL: https://ieeexplore.ieee.org/document/9499301.